УКРАЇНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЗАЛІЗНИЧНОГО ТРАНСПОРТУ

ФАКУЛЬТЕТ ІНОФОРМАЦІЙНО-КЕРУЮЧИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

Кафедра обчислювальної техніки та систем управління

МЕТОДИЧНІ ВКАЗІВКИ
до виконання лабораторних робіт

з дисциплін
«ОБЧИСЛЮВАЛЬНА ТЕХНІКА І ПРОГРАМУВАННЯ»
та *«ІНФОРМАТИКА»*

Частина 1

Харків – 2021

Методичні вказівки розглянуто і рекомендовано до друку на засіданні кафедри обчислювальної техніки та систем управління 26 квітня 2021 р., протокол № 9.

Методичні вказівки до виконання лабораторних робіт з дисциплін «Обчислювальна техніка і програмування» та «Інформатика» (частина 1) були розроблені для забезпечення виконавців робіт планом проведення лабораторних робіт та надання їм потрібного мінімуму теоретичного матеріалу, який необхідно знати. Завдання та описи лабораторних робіт дають змогу проводити роботи у двох формах: як у відповідних аудиторіях університету, так і дистанційно.

При виконанні завдань лабораторних робіт використовується об'єктно-орієнтована мова програмування високого рівня Microsoft Visual Basic Script Edition.

Методичні вказівки до виконання лабораторних робіт призначені для здобувачів вищої освіти будівельного факультету денної форми навчання за напрямами підготовки бакалаврів, спеціальності: 131 «Прикладна механіка», 133 «Галузеве машинобудування», 192 «Будівництво та цивільна інженерія», 193 «Геодезія та землеустрій», 273 «Залізничний транспорт».

Укладачі:

доценти С. Є. Бантюков,
В. Г. Пчолін

Рецензент

проф. В. І. Мойсеєнко

ЗМІСТ

ВСТУП

Методичні вказівки призначено для виконання лабораторних робіт у другому семестрі навчальної дисципліни «Обчислювальна техніка та програмування» студентами будівельного факультету.

Перед виконанням кожної роботи необхідно вивчити відповідний лекційний матеріал і звернути особливу увагу на загальні положення, що передують описам лабораторних завдань.

Наведені приклади програм слід розглядати лише як один із можливих варіантів розв'язання завдань.

Методичні рекомендації містять опис чотирьох лабораторних робіт з модуля «Мови програмування» із восьми робіт за курсом. Кожен розділ, що відповідає окремій лабораторній роботі, складається з таких підрозділів:

мета лабораторної роботи;

рекомендації щодо підготовки до виконання лабораторної роботи; основні теоретичні відомості, необхідні для її виконання;

загальна постановка завдання до лабораторної роботи та порядок дій при її виконанні;

приклади виконання завдань;

індивідуальні варіанти завдань.

При проведенні всіх лабораторних робіт використовується єдина конфігурація програмно-апаратних засобів: персональна ЕОМ типу IBM-PC з процесором не нижче Pentium III, операційна система Windows XP, Windows 7 або вище, інтернет-браузер Microsoft Internet Explorer версії 6 та вище або інший інтернет-браузер, що має кодову платформу Microsoft Trident.

Під час проведення лабораторних робіт студент повинен продемонструвати:

творчий, індивідуальний підхід до розроблення проектів (програмного коду);

грамотне використання наявного програмного забезпечення;

навички програмування алгоритмічною об'єкто-орієнтованою мовою високого рівня Microsoft Visual Basic Script Edition.

Студент повинен уміти перетворити свою програму в програмний продукт, використовувати якісний аналіз програми, виконувати оцінку отриманих результатів. Велике значення

водночас матиме вміння облаштовувати зручний інтерфейс з користувачем та застосовувати докладні коментарі до програми.

Варіант завдання до лабораторної роботи вибирається за домовленістю з викладачем або відповідно до номера прізвища студента в журналі групи.

1 Загальні вимоги до оформлення та змісту звіту лабораторної роботи

У разі виконання лабораторної роботи під час відповідного заняття в аудиторії навчального закладу в присутності викладача.

1 Звіт із лабораторної роботи виконується рукописно або в друкованому вигляді (комп'ютерний друк) на аркушах формату А4.

2 Звіт лабораторної роботи повинен починатися з титульного аркуша, на якому мають бути: назва дисципліни; тема лабораторної роботи; дата виконання роботи; ПІБ студента, курс, номер групи, номер варіанта індивідуальних завдань; ПІБ, посада викладача, що оцінюватиме звіт.

3 Опис виконання кожного завдання у звіті повинен починатися з докладного опису умови завдання.

4 Схеми алгоритмів у разі рукописного виконання креслити чорним олівцем або чорнилом за допомогою відповідних засобів для креслення. Написи до блоків робити розбірливо, бо інакше це розглядатиметься як помилка. У разі комп'ютерного друку для креслення можна застосовувати відповідне графічне програмне забезпечення.

5 Текст коду програми (HTML-документа) у разі рукописного виконання записувати друкованими літерами або застосовувати креслярський шрифт, водночас для відображення пробілів теж використовувати символ.

6 Результати розрахунків записувати так, як вони були виведені у вікні браузера.

7 Аркуші звіту з'єднувати скріпками або іншим загальноприйнятим способом.

У разі виконання лабораторної роботи дистанційно.

1 Звіт із лабораторної роботи виконується в електронному вигляді в одному загальному файлі чи в декількох файлах.

Для оформлення звіту одним файлом можна, наприклад, використати програмний засіб MS Word, щоб отримати файл у doc-форматі. У такому файлі текст, графіка, напівтонові та кольорові зображення вбудовуються як окремі об’єкти. Після попередньої згоди з викладачем як загальний файл можливо також застосовувати файли інших форматів.

Якщо звіт складається із декількох файлів, текстові дані, графіка (схеми) та зображення можуть зберігатися окремо у файлах відповідного формату, але на застосування кожного формату має бути згода викладача.

2 Звіт лабораторної роботи повинен мати титульні дані: назва дисципліни; тема лабораторної роботи; дата виконання роботи; ПІБ студента, курс, номер групи, номер варіанта індивідуальних завдань; ПІБ, посада викладача, що оцінюватиме звіт. У разі оформлення звіту одним файлом ці дані повинні розміщуватися початковим його компонентом. При звіті в декілька файлів їх розміщують в окремому текстовому файлі з назвою, наприклад, "title.txt".

3 Якщо звіт формується одним файлом, то в ньому опису виконання кожного завдання повинен передувати опис умови завдання, далі повинні йти результати виконання завдання у такій черзі, як вони вказуються у завданні до лабораторної роботи. У разі звіту в декілька файлів можливо опис умови кожного завдання та результати виконання кожного пункту кожного завдання розміщати в окремий файл відповідного формату з відповідним ім’ям, наприклад, "TaskCondition1-1.txt" (текстовий файл з описом умови завдання 1.1), "TaskProgramCode1-1.htm" (програмний код до завдання 1.1 у форматі HTML-документа) і т. д.

4 При отриманні результатів розрахунків треба фіксувати не тільки безпосередньо ці результати, але й «знімати» повні «скриншоти» під час виведення цих результатів на екран комп’ютерного пристрою, тобто отримувати растрові зображення всього екрана, на якому добре видно не тільки вікно з розрахунками, але й оточуючий фон. Ці зображення обов’язково повинні додаватися до звіту.

5 Для зручності пересилання звіту викладачеві файл (файли) звіту можна архівувати.

2 Правила розроблення сценаріїв мовою VBScript для розв’язання завдань розрахункового характеру

Виконання лабораторних робіт, описи яких наведено в цих методичних вказівках, пов’язано із розробками сценаріїв — комп’ютерних програм, що складені мовою програмування для інтерпретатора, тобто програмного засобу, який сприймає текст початкового коду як послідовність вказівок до виконання конкретних обчислень (як сценарій) і змушує процесор комп’ютерної системи їх виконувати. Метою всіх завдань, що пропонується виконати в лабораторних роботах, є отримання числового значення однієї деякої величини або набору значень деяких величин. Такі завдання, зазвичай, називають розрахунковими.

Для програмування процесів розв’язання завдань розрахункового характеру їх виконавець завжди повинен здійснити п’ять етапів дій:

1) визначити математичний метод (або декілька методів), що дає змогу (дають змогу) розв’язати при їх відповідному застосуванні завдання;

2) скласти алгоритм розв’язання завдання, який, зазвичай, записують у вигляді схеми алгоритму або за допомогою псевдокоду;

3) відповідно за алгоритмом скласти код програми, застосовуючи ту мову програмування, яку «розуміє» обраний обчислювальний засіб, що буде використаний безпосередньо як інструмент розв’язання завдання;

4) програмний код треба налагодити на контрольних прикладах, щоб той працював без помилок та збоїв;

5) використовувати (експлуатувати) цей програмний код для розв’язання потрібного завдання та інших подібних.

Ці етапи треба проходити та відображати у звіті при виконанні кожного завдання до кожної лабораторної роботи. Тобто кожне виконання завдання та опис його у звіті треба опрацьовувати за такими пунктами: 1) математична (формалізована) постановка завдання; 2) загальна схема алгоритму обчислення; 3) обчислення контрольних прикладів; 4) програмний код виконання завдання; 5) демонстрування результатів.

Розроблення схем алгоритмів обчислень треба здійснювати за встановленими правилами. Про них можна дізнатися із конспекту лекцій з дисципліни [1] або із навчального посібника [4].

Програмний код, що забезпечує виконання завдання, складається алгоритмічною мовою Microsoft Visual Basic Script Edition (VBScript), водночас інтерфейс користувача задається елементами веб-сторінок, для чого застосовується мова HTML, а увесь програмний код з указаними додаваннями оформлюється як HTML-документ для подальшого відображення в середовищі браузера. Усі необхідні відомості про розроблення програмного коду та його застосування можна дізнатися із конспекту лекцій [2].

Програмний код, що розміщується у звіті, повинен бути працездатним та правильно виконувати обчислення згідно з контрольними прикладами.

Програмний код, який щойно складений, як правило, не працює або працює неправильно, оскільки має помилки. Тому його треба налагодити.

Деякі версії MS IE мають програмні приладдя, що спрощують налагодження, але їх можливості малі. Існують спеціальні програмні засоби, але для їх застосування треба мати ліцензію. Нижче дається опис налагодження коду сценарію, при якому не застосовуються ніякі додаткові програмні засоби.

3 Опис методики налагодження коду

Налагодження сценарію — дуже відповідальний етап у всьому загальному процесі його створення. Без цього етапу завдання отримання скрипту, що вирішує ту чи іншу обчислювальну проблему, просто залишиться нерозв’язаним.

Ми вважатимемо, що скрипт подано в складі HTML-документа, щоб його можна було виконати за допомогою браузера IE. Якщо це так, то спочатку треба налагодити ту частину документа, яка безпосередньо складається із тегів мови HTML. Ця мова є описовою, вона служить для задавання зовнішнього вигляду веб-сторінки у вікні браузера. З її допомогою вказується, які і як елементи сторінки розташовуються один відносно іншого та які властивості й характеристики вони мають. Усе це задається шляхом вказівки послідовності

відповідних тегів та їх атрибутів. Тому налагодження зовнішнього вигляду сторінки полягає в поступовому нарощуванні записів тегів у HTML-документі з періодичним відображенням його у вікні браузера та корегуванням як атрибутів тегів, так і самих тегів у послідовності записів документа.

Таким чином, якщо із зображенням веб-сторінки у вікні браузера щось не так, то це означає, що помилка була зроблена саме при задаванні тегів HTML-сторінки. Тоді треба ретельно перевірити правильність задавання всіх компонентів тегів, зберігати зроблені зміни у файлі веб-сторінки і знову розгорнути сторінку в браузері. Цю процедуру слід повторювати доти, поки HTML-сторінка не стане відображатися у вікні браузера так, як було задумано.

Налагодження сценарію послідовно забезпечує загальну працездатність програми і правильність її обчислень. Попереднє зауваження: щоб почати налагодження сценарію, потрібно мати контрольні приклади, тобто набори значень вхідних величин, для яких відомі значення вихідних величин, стосовно конкретного алгоритму, який програмно реалізується.

Програмний код реалізації того чи іншого алгоритму, складений мовою VBScript, формально є послідовність рядків символів, які можна пронумерувати від 1 до деякого числа *n*. Але у разі багатомодульної структури сценарію така нумерація дасть змогу тільки ідентифікувати рядки, послідовність дій у скрипті може збігатися з нею лише у межах одного модуля. Тому розглянемо процес налагодження сценарію, коли той має один модуль (нехай це буде, наприклад, якийсь обробник події).

Помилки у сценаріях будемо розрізняти трьох видів: 1) синтаксичні помилки; 2) помилки, що пов’язані з конфліктами при обробці даних; 3) помилки загальнологічного характеру.

Помилки загальнологічного характеру, як правило, виникають у результаті невідповідності дій сценарію з обчисленнями алгоритму, з помилками в самому алгоритмі й навіть з невдалим вибором методу розв’язання задачі. Такі помилки виявляються при ретельному тестуванні сценарію на різних наборах контрольних прикладів. Прийоми їх виявлення ми тут не будемо розглядати.

Зупинимо увагу на перших двох видах помилок. Оскільки вони зупиняють розрахунки інтерпретатора VBS або заважають їм розпочатися.

Синтаксичні помилки — це помилки в ключах операторів, у записах виразів, операцій, імен змінних та констант, у результаті порушень мовних конструкцій тощо. При таких помилках інтерпретатор не може визначити порядок та характер обчислювальних дій і не починає їх виконання зовсім.

Помилки, що призводять до конфліктів обробки даних, — це помилки, що виникають при перетвореннях даних з типу в тип, при неможливості виконати ту чи іншу операцію з даними, що є, тощо. При таких помилках інтерпретатор, як правило, починає виконувати сценарій, але зупиняється, коли стикається з такими негараздами.

Отже, нехай веб-сторінка відобразилася в браузері правильно, у її текстові поля були внесені всі дані згідно з контрольним прикладом, але запуск скрипту не стався, ніби його в HTML-сторінці «не було», або щось було розраховано та виведено у текстові поля, у вікна повідомлень, але не все. Тобто сценарій явно до кінця потрібні розрахунки не зробив. Це є ознаки наявності в ньому помилок першого та (або) другого виду.

Тоді для налагодження коду пропонується виконати нижченаведені дії.

1 До сценарію відразу після тегу <SCRIPT> вставляється виклик вікна з якимось повідомленням, наприклад, ставиться інструкція:

MsgBox "Script start!".

Якщо цей рядок буде написано правильно, то він виведе відповідне вікно повідомлення на екран, як тільки скрипт почне виконуватися інтерпретатором.

Аналогічно після всіх інструкцій сценарію перед тегом </SCRIPT> ставимо ще один виклик вікна повідомлення, наприклад, інструкцію

MsgBox "Script stop!".

Якщо цей рядок було написано правильно, то поява вікна з відповідним повідомленням покаже, що скрипт виконався до кінця.

2 З цими доданими інструкціями завантажується до браузера HTML-документ сторінки. Тоді можливі три варіанти

подій. Варіант перший — обидва вікна з повідомленнями не з’явилися на екрані. Це є ознака, що у тексті сценарію (поміж тегами <SCRIPT> та </SCRIPT>) існує одна чи декілька синтаксичних помилок. Варіант другий — на екрані з’явилося тільки перше повідомлення: "Script start!". Це означає, що у тексті сценарію є одна чи декілька помилок, що ведуть обчислення до конфлікту даних. Варіант третій — на екрані з’явилися послідовно обидва повідомлення. Це може бути тоді, коли сценарій працює правильно або працює, але в його логіці виконання є помилка (про це свідчить некоректне виконання контрольного прикладу), або працює, але тільки тому, що інтерпретатор VBS не виконував той фрагмент сценарію, у якому є помилка конфлікту даних.

Нехай є перший або другий варіанти.

3 Перетворюємо всі рядки сценарію, що містяться поміж вставленими викликами вікон з повідомленнями, у коментарі. Для цього першим (самим лівим) символом у кожного з цих рядків ставиться апостроф. Зауважимо, що після такої процедури повторне відкриття сценарію у браузері призведе до послідовної появи вікон з повідомленнями, виклики яких було вставлено.

4 Послідовно проводимо зворотне перетворення рядків, що раніше були оголошені коментарями, у рядки дійсних інструкцій. Після кожного зворотного перетворення рядка перевіряється сценарій на працездатність. Якщо після чергового зворотного перетворення перестали виводитися обидва вікна, то в останньому рядку, що був перетворений на інструкцію, є синтаксична помилка. Якщо перестало виводитися тільки друге вікно (з повідомленням "Script stop!"), то в операторі, з якого останнім було знято знак коментарю, є конфлікт при обчисленні даних.

Коли рядок, де є помилка, знайдено, то, звісно, у ньому треба знайти саму помилку й виправити її. У разі синтаксичної помилки, зазвичай, достатньо, згадавши потрібні правила утворення операторів, ретельно переглянути рядок. Але, якщо цього замало, помилка не знайдена, то пропонується перенабрати рядок з клавіатури, ретельно стежачи за застосуванням регістрів. Такий прийом позбавить рядок від «кодової плутанини», якщо вона є. Якщо й цього недостатньо, то знову перевірити вміст

рядка з підвищеною увагою і наполегливістю. Адже помилка в рядку повинна бути!

У разі помилки конфлікту даних перед рядком, де є ця помилка, вставляють виклики вікон повідомлень, у які вписуються імена величин, що задіяні в інструкції рядка. Таким способом можна довідатися про значення всіх величин, що використані у конфліктному обчисленні, та шляхом логічного аналізу визначити помилку.

Зауважимо, що в процедурі зворотного перетворення коментарів на рядки інструкцій треба уважно стежити за зняттям апострофів з початкових рядків блокових операторів. У таких випадках разом зі зняттям апострофа з початкового рядка треба знімати апостроф і з кінцевого рядка того ж оператора, інакше це додасть до коду синтаксичну помилку. Тобто, якщо знято апостроф з рядка із ключем IF, то треба також знімати апостроф з рядка END IF; якщо знято апостроф з рядка із ключем FOR, то треба також знімати апостроф з рядка NEXT; якщо знято апостроф з рядка із ключем SUB, то треба також знімати апостроф з рядка END SUB і т. д.

Загальне зауваження. Нагадуємо, що спосіб налагодження коду, який наведено вище, добре працює при одномодульному сценарії. У разі багатомодульного сценарію таке налагодження коду дасть змогу тільки визначити рядки, де є синтаксичні помилки. Щоб знайти помилки, які викликані конфліктами обробки даних у багатомодульному сценарії, треба налагоджувати кожен модуль у сценарії окремо. Для цього в початок модуля (наприклад другий після рядка з ключем SUB) вставляється виклик вікна з повідомленням про початок, а в кінець модуля перед рядком з ключем END вставляється виклик вікна з повідомленням про завершення модуля. Інші модулі сценарію та фрагменти головного модуля, які не пов’язані із викликом та забезпеченням даними модуля, що налагоджується, «закривають» апострофами. Далі відтворюється процес налагодження, подібний тому, що був описаний. Так послідовно налагоджуються всі модулі сценарію.

Звісно, що можна використовувати й інші прийоми налагодження коду. Наприклад, не «закривати» код шляхом перетворення його у коментар, а послідовно переписувати його

рядок за рядком у файл нового HTML-документа і далі вже тестувати цей новий файл. Але головна сутність процесу налагодження, яка полягає в поступовій перевірці кожного рядка коду від початку модулів і до їх кінця, повинна зберігатися.

ЛАБОРАТОРНА РОБОТА 1. Підготовка і розв'язання на ПЕОМ задач за лінійними алгоритмами

Мета лабораторної роботи – отримання практичних навичок підготовки, налагодження та виконання лінійних програм, складених алгоритмічною мовою VBScript.

1.1 Завдання лабораторної роботи

1 Вивчити теоретичний матеріал, який надається в підрозділі 1.2, та матеріали, на які вказують посилання із цього підрозділу.

2 Виконати відповідно до наведеної умови завдання 1.1 за своїм варіантом. Для цього треба: 1) скласти схему алгоритму розв'язання завдання (за вимогою викладача); 2) скласти код HTML-документа, до якого вбудовано код сценарію, що заданий мовою VBScript і який виконує обчислення завдання 1.1. Введення-виведення даних у ньому забезпечити через стандартні функції VBS InputBox та MsgBox; 3) скласти самостійно контрольні приклади, за допомогою яких можна перевірити працездатність та правильність роботи програми; 4) налагодити код; 5) зробити розрахунки й зафіксувати результати.

3 Виконати відповідно до наведеної умови завдання 1.2 за своїм варіантом.

4 Скласти звіт про результати виконання лабораторної роботи згідно з вимогами до змісту звіту, що наведені у розділі 1. Зауважимо, що у звіті мають бути відображені результати виконання кожного пункту кожного завдання.

5 Передати звіт викладачу для оцінювання, домовитися з ним про процедуру захисту лабораторної роботи та здійснити захист.

1.2 Довідка про теоретичні засади, які потрібні при виконанні роботи

Перед виконанням лабораторної роботи її виконавець повинен знати:

класифікацію базових підтипів загального типу даних мови VBScript і їхні основні характеристики;

лексичні основи мови VBScript – поняття: змінна, вираз, операнд, константа, оператор;

пріоритети операцій;

правила перетворення підтипів;

основні бібліотечні математичні функції мови VBScript.

Ці теоретичні матеріали наведено у конспекті лекцій [2, с. 7–26], у розділах: «Типи даних у мові VBScript», «Складання програм мовою VBScript», «Застосування змінних і констант у мові VBScript» та «Запис обчислень у VBS та оператор присвоєння».

Для забезпечення введення та виведення даних між користувачем та середовищем обчислень у цій роботі треба застосовувати відповідні стандартні функції мови VBScript: MsgBox та InputBox. Докладний опис їх застосування наведено у розділі «Найпростіший спосіб введення і виведення даних у сценаріях VBScript» конспекту лекцій [2, с. 26–31].

Слід зауважити, що функція MsgBox є не тільки «найпростішим способом» виведення даних, але й облаштовує зручність процесу налагоджування сценаріїв VBScript (дивись розділ 3).

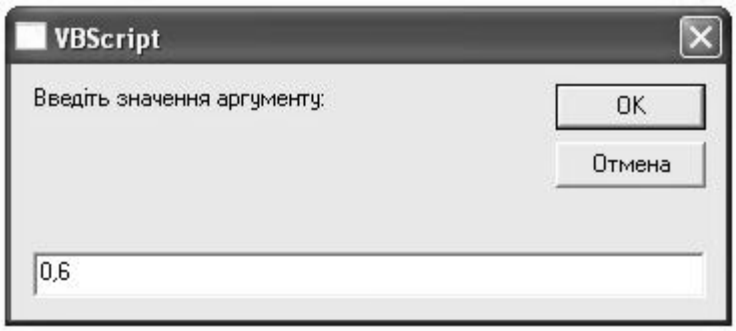
Для «швидкого» застосування функцій MsgBox та InputBox достатньо задати у їх викликах тільки по одному аргументу, яким, звісно, повинен бути символьний рядок. Водночас слід пам’ятати, що запис аргументу при виклику функції InputBox потрібно брати в круглі дужки. Допоміжних відомостей вікна, які відкриватимуться при таких викликах, мати не будуть, але зміст символьних рядків, що є значеннями аргументів цих функцій, виводитиметься завжди коректно. Наприклад, якщо сценарій VBS, що вкладений у HTML-документ, складається лише з одного рядка:

MsgBox "Значення арктангенса буде таке: " & _

Atn(InputBox("Введіть значення аргументу:")) & "." ,

то при його виконанні послідовно на екрані монітора з’являться два вікна (рисунок 1.1). У першому вікні користувачеві буде запропоновано ввести числове значення аргументу для обчислення функції арктангенса (рисунок 1.1, а, користувач увів значення "0,6"), а в другому вікні буде наведено результат обчислення (рисунок 1.1, б).

а)



б)

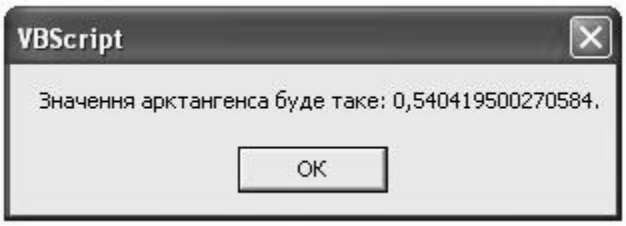


Рисунок 1.1 – Зразок зовнішнього вигляду вікон, що з’являються при використанні функцій InputBox та MsgBox відповідно

Загальний процес створення сценарію лінійного обчислення наведено у розділі «Програмування лінійного обчислювального процесу засобами VBScript» конспекту лекцій [2, с. 31–32]. Як запровадити сценарій VBScript до складу HTML-документа, щоб мати змогу виконати його у програмному середовищі браузера IE, докладно надано в розділі «Виконання сценарію VBS за допомогою браузера MS Internet Explorer» конспекту лекцій [2, с. 32–36].

1.3 Розгляд основних пунктів виконання завдання 1.1

Приклад варіанта завдання 1.1: розробити сценарій VBScript лінійного обчислення згідно із записом:

$$y = \frac{a}{b} + \frac{\sin^2 a - 8}{\ln b} \; ; \; b = \frac{a^2 - c}{2} \; ; \; c = 1{,}3 \; .$$

При введенні та виведенні даних використовувати вбудовані функції MsgBox та InputBox.

Основну увагу при розробленні алгоритму лінійного обчислення треба звертати на задавання порядку послідовності розрахунків проміжних величин. Так, для наведеного обчислення треба спочатку у розрахунковому просторі задати величину *c*, потім визначити значення величини *b* і тільки наприкінці розрахувати *y*. Тобто схему алгоритму можна задати такою, яка наведена на рисунку 1.2.

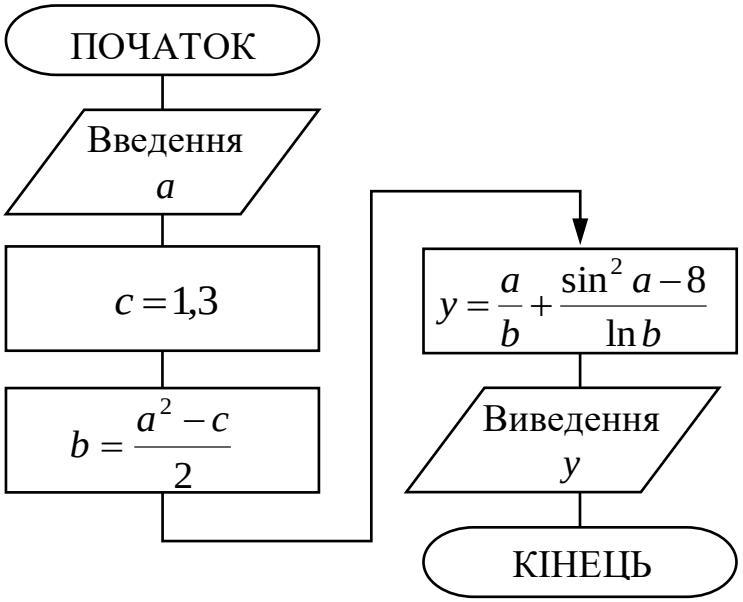


Рисунок 1.2 – Схема алгоритму розв’язання прикладу завдання 1.1

Код сценарію цього обчислення, введений до складу HTML-документа, такий:

<HTML> <HEAD>
<TITLE>Перша лінійна програма</TITLE>
<SCRIPT LANGUAGE="VBScript"> <!--
DIM a, b, c, y
a = 0 + InputBox("Введіть значення для змінної a. ", "Вікно вводу")

```
c = 1.3
b = ( a ^ 2 - c) / 2
y = a / b + (Sin(a) ^ 2 - 8) / Log(b)
MsgBox "Значення змінної y буде таке: " & y, , "Вікно виводу"
--> </SCRIPT> </HEAD> <BODY>
</BODY> </HTML>
```

Для перевірки правильності роботи коду слід розрахувати окремо (без застосування цього сценарію) контрольні приклади. Для цього задається деяке конкретне значення величини *a* і далі послідовно обчислюються значення для величин *b* та *y* згідно із записом їх обчислення із умови завдання. При обчисленні можна застосовувати будь-які калькулятори (наприклад, програму «Калькулятор» із допоміжного забезпечення ОС Window), електронні таблиці та інші програмні й апаратні засоби для розрахунків.

У цьому прикладі варіанта завдання величини *a* може набувати не кожне числове значення, оскільки вона пов’язана з величиною *b*, а остання може бути тільки величиною додатною. Але, щоб не порушувати лінійну структуру сценарію, ніяких перевірок коректності вхідних даних не робилося. Якщо в інших варіантах завдання 1.1 виникатимуть подібні випадки, також не треба робити аналіз та ускладнювати код.

Завдання 1.1

Скласти сценарій лінійного обчислення мовою VBS та перевірити його роботу на контрольних прикладах.

Зазвичай, коли задаються подібні математичні викладки, то їх записують у тій послідовності, у якій вони повинні обчислюватися; але це не є обов’язковою вимогою для подібних записів. Що й підтверджують наведені варіанти завдання (таблиця 1.1). Будьте уважні при його виконанні!

Таблиця 1.1 – Варіанти завдання 1.1

Варіант	Завдання лінійного обчислення
1	$q=\dfrac{2x\sqrt{x^3}+\ln 2x+2x}{a+mc+t^2};\ t=\dfrac{mb}{a+x};\ m=a+2x;\ b=4{,}2$
2	$t=\dfrac{2\sin s+m^2}{a^2-x}+\sqrt{q+c^3};\ s=\dfrac{a+2c}{\ln m};\ m=2\sqrt{a^2+c};\ a=5{,}2$
3	$y=\dfrac{cx^2+m\cos b^3+\ln^2 x}{b+m+12};\ m=2x+\sqrt{x};\ b=ac^2-m$
4	$r=\dfrac{mc+\sqrt{2x}}{\sin^2 3x+2f};\ f=\dfrac{a}{x^3+b}+\sin\dfrac{m}{2c};\ m=b+\cos(a-x)$
5	$z=(a+bx)/m+c/(a+x)-q;\ q=\sin[(b+cx)/2];\ a=bx-m$
6	$c=(a+\sin 2x)/(m-r);\ r=2m/x+\operatorname{arctg}2x;\ m=(b+2x)/a$
7	$m=2a-cd;\ f=\dfrac{3y}{m+\sin 2x}+\sqrt{\dfrac{b}{m+2x}};\ d=c+x^2\sin 2x$
8	$q=(m+x)/\ln^2 x;\ r=\sqrt{t+2q};\ m=(c-2d)/(a+x);$ $c=\sqrt{x^3}$
9	$s=7/(m+2t)+a/(b+cx)-4z;\ m=\sqrt{a+cx^2};$ $b=3{,}25;\ \ z=3m+t^2$
10	$z=2x/(b+x)+c/(b-y)+2bx/(b-xy);\ y=2\ln\lvert x-a\rvert;$ $b=c-2x\sin x;\ c=12{,}5$
11	$q=mc^2+2f^3+2\cos^2 x;\ m=a/c+b/x+\sqrt{f};$ $a=3{,}75;\ f=\ln(a+2{,}5x)$
12	$z=m^2cx/\sqrt{a}+bc^3-\ln^2 a;\ b=a+\sin 2x^2;\ a=2{,}25;$ $m=a+b^2-\cos x$
13	$c=b+2x-\sin x;\ r=(c+2x^2)/m+at^2+\sqrt{x};\ a=3{,}25;$ $m=(2+x)/c+\operatorname{arctg}x-a$
14	$q=3m/(ax)+\sqrt{b+cx}+\lg\lvert b-mx\rvert;m=a+r^3+\cos^2 2x;$ $c=2a-x;\ r=c+2x$
15	$q=(at^2+rc-ax)/(b+\sqrt{c+x}-5);\ r=(mx+2t+c)^3;$ $m=2c^2x;\ t=ax^2+bx-c;\ c=7{,}5$

Завдання 1.2

Нехай $y = f(x)$, то для знаходження похідної y' в точці x використовується відомий вираз

$$y' = \frac{df(x)}{dx} = \frac{f(x+\Delta x) - f(x)}{\Delta x}.$$

Скласти сценарій мовою VBS для обчислення похідної функції $f(x)$ у заданій точці за наведеною наближеною формулою згідно з отриманим варіантом функції (таблиця 1.2). Для змоги оцінки точності обчислення додайте також до того ж сценарію обчислення похідної за аналітичним виразом, що додається до кожного варіанта функції. Введення та виведення даних зробіть за допомогою вбудованих функцій InputBox та MsgBox.

Таблиця 1.2 – Варіанти завдання 1.2

Варіант	Функція	Похідна	Δx
1	2	3	4
1	$y = a^5 + 5a^3x^2 - x^5$	$y' = 10a^3x - 5x^4$	1.0e-11
2	$y = \frac{ax+b}{a+b}$	$y' = \frac{a}{a+b}$	0.5e-10
3	$y = (x-a)(x-b)$	$y' = 2x - (a+b)$	1.0e-11
4	$y = (x+1)(x+2)^2(x+3)^3$	$y' = 2(x+2)(x+3)^2 \times \times (3x^2 + 11x + 9)$	0.5e-10
5	$y = (x\sin a + \cos a) \times \times (x\cos a - \sin a)$	$y' = x\sin 2a + \cos 2a$	1.0e-11
6	$y = (1 + nx^m)(1 + mx^n)$	$y' = mn[x^{m-1} + x^{n-1} + + (m+n)x^{m+n-1}]$	0.5e-10
7	$y = (1-x)(1-x^2)^2(1-x^3)^3$	$y' = -(1-x)^2(1-x^2)(1-x^3)^2 \times \times (1 + 6x + 15x^2 + 14x^3)$	1.0e-11

Продовження таблиці 1.2

1	2	3	4
8	$y=\frac{1}{x}+\frac{2}{x^2}+\frac{3}{x^3}$	$y'=-\left(\frac{1}{x^2}+\frac{4}{x^3}+\frac{9}{x^4}\right),$ $x\neq 0$	0.5e-10
9	$y=\frac{ax+b}{cx+d}$	$y'=\frac{\begin{vmatrix} a & b \\ c & d \end{vmatrix}}{(cx+d)^2}$	1.0e-11
10	$y=\frac{2x}{1-x^2}$	$y'=\frac{2(1+x^2)}{(1-x^2)^2}, \quad \lvert x\rvert\neq 1$	0.5e-10
11	$y=\frac{1+x-x^2}{1-x+x^2}$	$y'=\frac{2(1-2x)}{(1-x+x^2)^2}$	1.0e-11
12	$y=\frac{x}{(1-x)^2(1+x)^3}$	$y'=\frac{1-x+4x^2}{(1-x)^3(1+x)^4},$ $\lvert x\rvert\neq 1;$	0.5e-10
13	$y=\cos 2x-2\sin x$	$y'=-2\cos x(1+2\sin x)$	1.0e-11
14	$y=(2-x^2)\cos x+2x\sin x$	$y'=x^2\sin x$	0.5e-10
15	$y=\sin(\cos^2 x)\cdot\cos(\sin^2 x)$	$y'=-\sin 2x\cdot\cos(\cos 2x)$	1.0e-11

ЛАБОРАТОРНА РОБОТА 2. Підготовка і розв’язання на ПЕОМ задач за розгалуженими алгоритмами

Мета лабораторної роботи – отримання практичних навичок підготовки, налагодження та виконання розгалужених програм, складених алгоритмічною мовою VBScript.

2.1 Завдання лабораторної роботи

1 Вивчити теоретичний матеріал, який надається в підрозділі 2.2, та матеріали, на які вказують посилання із цього підрозділу.

2 Виконати відповідно до наведеної умови завдання 2.1 за своїм варіантом. В умові завдання пропонується при складанні сценарію обчислення скористатися як оператором умовного

переходу, так і оператором вибору. Останній треба застосовувати так, як показано в підрозділі 2.2.

3 Виконати відповідно до наведеної умови завдання 2.2 за своїм варіантом. Обов’язково складати код сценарію шляхом розроблення початкового фрагмента коду для перевірки одного числа із визначеного за теоремою набору чисел, які можуть бути коренями рівняння, та подальшого копіювання цього фрагмента із заміною значення вказаного числа на наступне із набору, намагаючись побудувати працездатний і правильний код якомога швидше.

4 Скласти звіт про результати виконання лабораторної роботи згідно з вимогами до змісту звіту, які наведено у розділі 1.

2.2 Довідка про теоретичні засади, які потрібні при виконанні роботи

Перед виконанням лабораторної роботи її виконавець повинен знати:

теоретичні відомості, які використовувалися при виконанні попередньої лабораторної роботи;

методику розроблення сценаріїв із загальною лінійною частиною й декількома гілками;

алгоритми виконання й синтаксис операторів If...Then...Else та Select Case.

Теоретичні матеріали стосовно двох останніх пунктів із наведеного списку розглядаються у конспекті лекцій [2, с. 55–59] в розділі «Завдання розгалужених обчислювальних процесів в сценаріях VBScript».

Найчастіше для задавання розгалужених обчислювальних процесів у сценаріях VBScript застосовують оператор умовного переходу, який ще іноді називають згідно з його ключем — "оператор If...Then...Else". Другим за частотою використання при створенні розгалужень у коді скриптів застосовують оператор вибору (інша його назва згідно з ключем — "оператор Select Case").

Оператор If...Then...Else вживається у скриптах VBS у двох формах синтаксису: лінійній (рядковій) і блоковій.

Лінійна форма оператора умовного переходу обов’язково

повинна бути задана «одним рядком», тому вона застосовується, як правило, у випадках, коли треба для зміни послідовності дій сценарію перевіряти виконання лише однієї умови. З цієї ж причини ця форма оператора часто застосовується в скороченому вигляді (без ключа Else).

Блокова форма може задавати послідовно перевірку декількох умов і таким чином забезпечувати керування будь-яким розгалуженим процесом обчислень.

Але, щоб уміло користуватися обома формами цього оператора, треба добре знати алгоритм дій інтерпретатора VBScript при їх виконанні. Це не тільки запобігатиме помилкам при задаванні порядку в обчисленнях, але й може спрощувати задавання умов виконання розгалужень.

Підтвердимо це на прикладі складання скрипту обчислення за формулою

$$y = \begin{cases} 1, \text{якщо } -0,5 < x < 2,5, \\ 2, \text{якщо } 2,5 \le x, \\ 3 \text{ в інших випадках}, \end{cases}$$

при додатковій вимозі — не застосовувати в сценарії логічні вирази та відношення.

Якби не додаткова вимога, то задане обчислення реалізує достатньо простий фрагмент сценарію VBS, у якому застосовується блоковий оператор умовного переходу:

```
IF x > -0.5 And x < 2.5 THEN
    y = 1
ELSEIF x >= 2.5 THEN
    y = 2
ELSE
    y = 3
END IF
```

Але й при виконанні додаткової вимоги можна застосувати блоковий оператор умовного переходу, оскільки як умови, що використовуються у його конструкції, дозволяється задавати арифметичні вирази. Якщо арифметичний вираз при обчисленні дає якесь числове значення, що відрізняється від нуля, то це прирівнюється до виконання умови; якщо — дорівнює нулю, то

вважається, що умова не виконується.

Такий варіант фрагмента скрипту також реалізує розглянуту формулу

```
IF ( 1.5 - Abs(x - 1) + Abs( 1.5 - Abs(x - 1))) /3 THEN
    y = 1
ELSEIF x + Abs(x) THEN
    y = 2
ELSE
    y = 3
END IF
```

Цей фрагмент коду не має жодної логічної операції чи операції відношення. Але він і попередній фрагмент роблять однакові обчислення.

Пояснення. Арифметичний вираз, що записаний між ключами IF та THEN, є задаванням функції, яку іноді називають функцією зубця:

$$f(x)=\frac{1}{3}\left(1{,}5-\left|x-1\right|+\left|1{,}5-\left|x-1\right|\right|\right).$$

На рисунку 2.1 наведено її графік.

Рисунок 2.1 – Графік функції зубця

А вираз, обчислення якого в коді увійшло до третього рядка між ключами ELSEIF та THEN, можна уявити як розрахунок функції $f(x)=x+\left|x\right|$, графік якої має такий вигляд, як на рисунку 2.2.

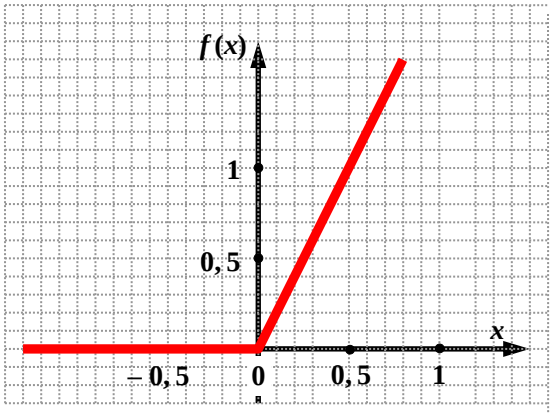


Рисунок 2.2 – Графік функції $f(x) = x + |x|$

Ці два арифметичні вирази «працюють» в операторі умовного переходу разом як один «логічний фільтр». Вираз першого рядка скрипту дає результатом обчислення нуль (а це означає — "False") при значеннях *x*, які дорівнюють 2,5 та більше або які дорівнюють – 0,5 та менше. Тобто тільки при цих значеннях *x* буде обчислюватися арифметичний вираз із умови третього рядка фрагмента скрипту, що розглядається. А вираз цей для всіх додатних значень *x* дає ненульовий результат обчислення (тобто "True"). Але додатні значення *x* , які можуть у нього потрапити для обчислення, будуть тільки ті, що дорівнюють 2,5 та більше, інші не будуть допущені фільтром з першого рядка фрагмента скрипту. Таким чином, другий варіант фрагмента сценарію даватиме такий самий результат обчислення, що й перший варіант фрагмента при однакових вхідних значеннях *x*.

Другим засобом організації розгалужених обчислень у мові VBScript є оператор вибору. Він дає змогу виконувати ті чи інші дії залежно від множини «дозволених» значень деякого заданого тестового виразу. У складі мовних конструкцій Basic він існує теж із ранніх діалектів останньої, але у VBScript він має додаткові можливості: у тестовому виразі та у списках виразів, що розташовані в рядках із ключами Case, можна використовувати будь-які обчислювальні вирази та константи. Це забезпечує у VBS використання операторів вибору там, де зазвичай використовувалися оператори умовного переходу. Наприклад, формулу обчислення величини *y*, що наводилась

вище, можна також реалізувати таким фрагментом сценарію мовою VBScript:

```
SELECT CASE True
    CASE x > -0.5 And x < 2.5  : y = 1
    CASE x >= 2.5   : y = 2
    CASE ELSE : y = 3
END SELECT
```

Але, звісно, виключити із цього запису логічні вирази та відношення вже не вдасться.

2.3 Розгляд основних пунктів виконання завдання 2.1

Нехай умовою варіанта такого завдання буде розробка та налагодження сценарію, складеного мовою VBScript розгалуженого обчислення, яке задано такими аналітичними виразами:

$$z = \begin{cases} ab - y^2, \text{ якщо } a > b \text{ та } b > y, \\ 2b, \text{ якщо } b > a \text{ або } y > b, \\ 3a, \text{ в інших випадках;} \end{cases} \qquad y = \begin{cases} (a-3)^2, \text{ при } x > 0, \\ 2a, \text{ при } x = 0, \\ 5, \text{ в інших випадках.} \end{cases}$$

З цієї умови зрозуміло, що обчислити величину *z* не можна, якщо відсутнє значення величини *y*, тому послідовність обчислень величин є така: спочатку обчислюється значення для величини *y*, потім — значення для *z*. Усі інші величини, що входять до виразів, є величини вхідні. Структури завдань обчислень величин *z* та *y* є багатоваріантні з умовами вибору розрахунків, тобто саме для їх програмної реалізації слід застосувати конструкції розгалужень. Зважаючи на ці міркування, складається схема алгоритму обчислення завдання, що наведена на рисунку 2.3.

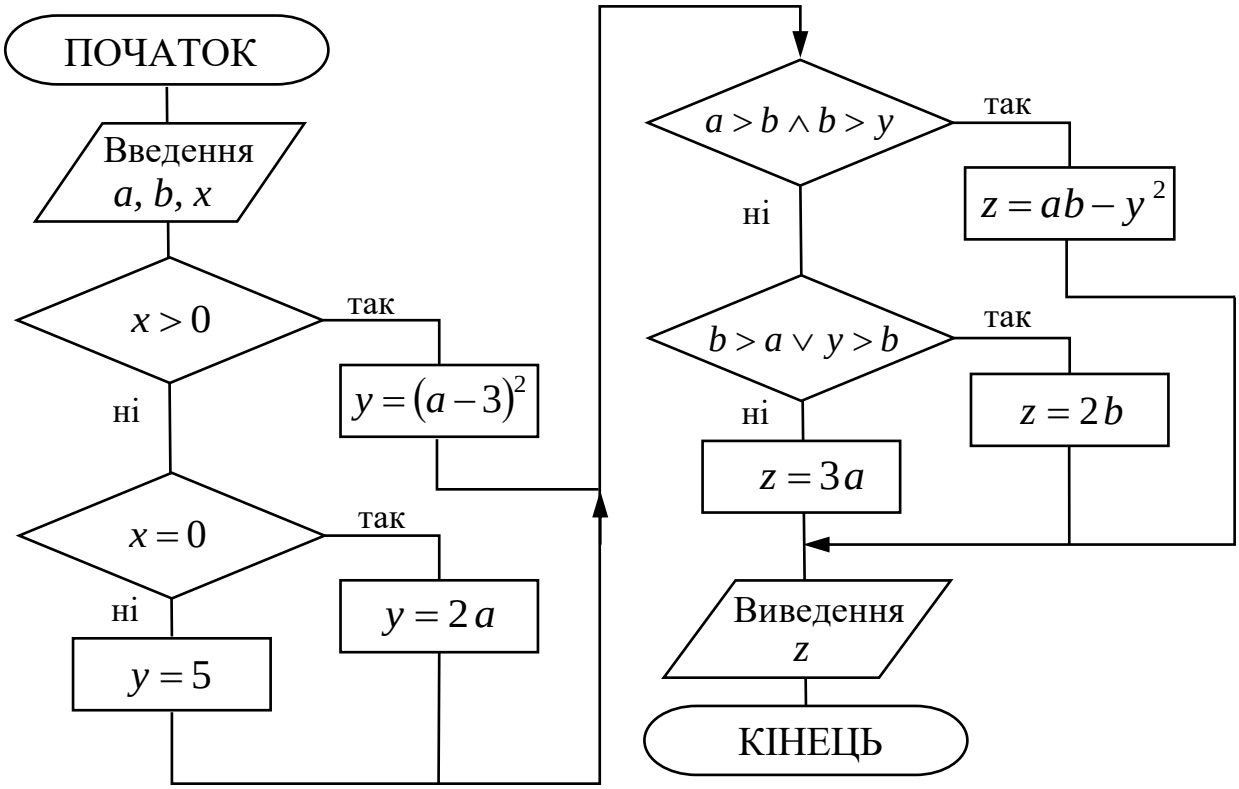


Рисунок 2.3 – Схема алгоритму розв’язання прикладу завдання 2.1

Згідно із загальними рекомендаціями до виконання варіантів завдання 2.1 бажано, щоб при складанні сценарію були застосовані обидва оператори організації розгалужень: If...Then...Else та Select Case. Для цього прикладу варіанта завдання немає різниці для обчислення якого з аналітичних виразів застосовувати той чи інший оператор. Щодо застосування лінійної форми оператора умовного переходу, то в цьому випадку її використання зовсім не має сенсу. Тому код сценарію потрібного обчислення, що введений до складу HTML-документа, може бути такий:

```
<HTML> <HEAD>
<TITLE> Виконання завдання 2.1, варіант N</TITLE>
</HEAD> <BODY>
<SCRIPT LANGUAGE="VBScript"> <!--
DIM a, b, x, z
x  = 0 + InputBox ("Введіть значення для змінної x. ",
"Вікно вводу")
```

```
a  = 0 + InputBox ("Введіть значення для змінної a. ",
"Вікно вводу")
b  = 0 + InputBox ("Введіть значення для змінної b. ",
"Вікно вводу")
IF x > 0 THEN
    y = (a -3 ) ^ 2
ELSEIF x = 0 THEN
    y = 2 * a
ELSE
    y = 5
END IF
SELECT CASE True
    CASE a > b And b >  y : z = a * b - y ^ 2
    CASE b > a Or y > b  : z = 2 * b
    CASE ELSE : z = 3 * a
END SELECT
MsgBox "Значення змінної z буде таке: " & z, , "Вікно
виводу"
--> </SCRIPT> </BODY> </HTML>
```

Контрольні приклади застосовуються при налагодженні коду. Якщо вони розробляються для налагодження сценаріїв з розгалуженими обчисленнями, то їх треба складати таким чином і в такій кількості, щоб кожна гілка обчислення була перевірена. Це є іноді досить складним завданням. Тому часто налагодження здійснюється окремо для лінійних частин обчислень і окремо для моделі структури розгалужень, тобто застосовуються ті самі прийоми, що й у разі налагодження багатомодульних сценаріїв.

Завдання 2.1

Треба скласти скрипт мовою VBS розгалуженого обчислення згідно з варіантом індивідуального завдання (таблиця 2.1).

При виконанні завдання бажано використовувати як оператор If...Then...Else, так і оператор Select Case. Лінійну форму оператора If...Then...Else використовуйте там, де вона, дійсно, буде зручною у застосуванні. Слідкуйте за послідовністю обчислення виразів!

Таблиця 2.1 – Варіанти завдання 2.1

Варіант	Завдання розгалуженого обчислення
1	2
1	$q=\begin{cases}2a^2-r^x, \text{ якщо } 0\le x<5 \text{ та } r\ne 0;\\ \sqrt{x+r}\ , \text{ якщо } 5\le x<10 \text{ або } a>2x;\\ a+x+r, \text{ в інших випадках;}\end{cases}$ $r=x^3-1+\sqrt{a-1}\,;$ $s=\begin{cases}2q, \text{ якщо } q>12{,}7;\\ 3q, \text{ якщо } q\le 12{,}7\end{cases}$
2	$y=\begin{cases}x^2+a+1, \text{ якщо } 0<x<1{,}5;\\ 2\ln x-a, \text{ якщо } 1{,}5\le x<3 \text{ та } a>0;\\ \sqrt{\lvert a\rvert+x}, \text{ якщо } 3\le x<5;\\ 12x^2, \text{ в інших випадках;}\end{cases}$ $z=\begin{cases}3y, \text{ якщо } y\le 3{,}8\ ;\\ 7y, \text{ якщо } y>3{,}8\end{cases}$
3	$s=\begin{cases}2x^m-\ln c^2, \text{ якщо } x^m<396 \text{ та } c>5;\\ x^5-\lg m, \text{ якщо } x^m=396 \text{ або } c\le 2;\\ c^5+\sqrt{x}, \text{ якщо } x^m>458;\\ b+3mx+x^m, \text{ в інших випадках;}\end{cases}$ $z=5{,}2+\lg q-2x;$ $q=\begin{cases}\ln\dfrac{s}{m}+x^{m+1}, \text{ якщо } m>5;\\ 10, \text{ в інших випадках}\end{cases}$
4	$t=\begin{cases}x^2-am^2, \text{ якщо } c>x>a \text{ та } b>5;\\ cx^5+2b, \text{ якщо } x\le a \text{ або } b=3;\\ \sqrt{cx^2+b}, \text{ якщо } x=c;\\ 0, \text{ в інших випадках;}\end{cases}$ $z=3{,}5t+2;$ $q=\begin{cases}z+2t, \text{ якщо } z\le 39\ ;\\ \ln(z+t), \text{ якщо } z>39\end{cases}$
5	$r=\begin{cases}a+2x, \text{ якщо } x<0 \text{ та } a>2;\\ b-cx, \text{ при } x\ne 4, \text{ якщо } x>0 \text{ або } a=0;\\ a^2-2\sqrt{x}, \text{ якщо } x=4 \text{ та } c=7;\\ \sqrt{b+x}, \text{ в інших випадках;}\end{cases}$ $s=\begin{cases}2r+a^x, \text{ якщо } r\ge 7{,}2;\\ 3r-x, \text{ якщо } r<7{,}2\end{cases}$
6	$y=\begin{cases}x+2a^m, \text{ якщо } a+m=4{,}6 \text{ та } x\ge 7;\\ \sqrt{x}+3a, \text{ якщо } a+m\ge 5{,}2\ ;\\ \ln\left(a+x^m\right), \text{ в інших випадках}\ ;\end{cases}$ $q=a+2y-3z;$ $z=\begin{cases}2y, \text{ якщо } y<16{,}5\ ;\\ 5y-c, \text{ якщо } y\ge 16{,}5\end{cases}$
7	$y=\begin{cases}cx^3+8a, \text{ якщо } x=5 \text{ або } b+c<12\ ;\\ \ln(x-a), \text{ якщо } c\le x<b\ ;\\ b+3c, \text{ в інших випадках}\ ;\end{cases}$ $r=\begin{cases}3y, \text{ якщо } y>15\ ;\\ a-y, \text{ якщо } y=15\ ;\\ b+2x, \text{ якщо } y<15\end{cases}$

Продовження таблиці 2.1

1	2
8	$z=\begin{cases} ax^3+b\ln 2x^2, \text{ якщо } x>a \text{ та } b=7; \\ c\sqrt{\lvert b+cx^2-a\rvert}, \text{ якщо } x<a; \\ \lg\left(2x+3b^3\right), \text{ в інших випадках;} \end{cases}$ $c=\dfrac{x}{b}+\dfrac{b}{a};$ $q=\begin{cases} 2z^2+\ln a, \text{ якщо } z>153; \\ 3z^3+\mathrm{e}^x, \text{ якщо } z\le 153 \end{cases}$
9	$f=\begin{cases} a+mx, \text{ якщо } x<0; \\ b+cx^2-a, \text{ якщо } 0\le x<2 \text{ та } a>b; \\ 3x+2b, \text{ якщо } 2\le x<5; \\ x^2, \text{ в інших випадках;} \end{cases}$ $c=2a+\ln^2 2x;$ $s=\begin{cases} 2f, \text{ якщо } f<7; \\ 5f, \text{ якщо } f>7; \\ 3f, \text{ якщо } f=7 \end{cases}$
10	$r=\begin{cases} bx^2+\ln 2x+\sqrt{x}, \text{ при } c<8, \text{ якщо } x\ge 5 \text{ або } b=3; \\ \lg^2\lvert x-a\rvert+b\dfrac{c-x}{a-m}, \text{ в інших випадках;} \end{cases}$ $q=\begin{cases} r+\operatorname{arctg} x, \text{ якщо } r=56{,}4; \\ \mathrm{e}^{2x}+\sqrt{\lvert r\rvert-a}, \text{ якщо } r>56{,}4; \\ \sin^2 2x+3r-b\sqrt{cx}, \text{ якщо } r<56{,}4 \end{cases}$
11	$t=\begin{cases} a+2x^c, \text{ якщо } x+c=2{,}9 \text{ та } c<5; \\ m-3b^2, \text{ якщо } x+c>2{,}9 \text{ або } 1\le m\le 4; \\ c+\ln x, \text{ в інших випадках;} \end{cases}$ $q=\dfrac{x}{t}+\dfrac{t}{2m}+\sqrt{a+t};$ $s=\begin{cases} a-t, \text{ якщо } t>q; \\ b+t, \text{ якщо } t\le q \end{cases}$
12	$r=\begin{cases} cx^2+\sqrt{b}-c, \text{ якщо } 2<x\le 5 \text{ або } b>25; \\ c\sin^2 x+10, \text{ якщо } 8<b\le 25; \\ \sqrt{cx}+a, \text{ якщо } 0<x\le 2 \text{ та } c=3; \\ b-2ac, \text{ в інших випадках;} \end{cases}$ $q=3r+2bx;$ $t=\begin{cases} 2q, \text{ якщо } q<12{,}8; \\ 3r+10, \text{ якщо } q=12{,}8; \\ q-3a, \text{ якщо } q>12{,}8 \end{cases}$
13	$f=\begin{cases} cx^2+\ln x, \text{ якщо } 2<x<4; \\ b-cx, \text{ якщо } 4\le x<6; \\ a^3+\lg x, \text{ якщо } 6\le x<8; \\ cx+b^x, \text{ якщо } x\le 2 \text{ або } x\ge 8; \end{cases}$ $a=\dfrac{\sin^2 x}{b}+\dfrac{x^2}{\ln x};$ $s=\begin{cases} 2f, \text{ якщо } f=14{,}5; \\ 3f, \text{ якщо } f\ne 14{,}5 \end{cases}$
14	$y=\begin{cases} 3x+b\sin x, \text{ якщо } 2<x<4 \text{ та } b<8; \\ \dfrac{5x^3}{a^2}-a\cos 2x, \text{ якщо } 4\le x<9; \\ 7x+2ab, \text{ в інших випадках;} \end{cases}$ $z=3y+\dfrac{2ax}{b}-\sqrt[3]{y};$ $r=\begin{cases} z+y, \text{ якщо } z>25; \\ zy, \text{ якщо } 20\le z\le 25; \\ y-z, \text{ якщо } z<20 \end{cases}$

Продовження таблиці 2.1

1	2
15	$z=\begin{cases}\dfrac{\sqrt{b}}{\sin b}+\ln x-2, \text{ якщо } bx>9 \text{ та } a\neq 2;\\ \sqrt[3]{cx+5a-c}, \text{ якщо } 6<bx\le 9;\\ \ln\left(b+ax^2\right)-\dfrac{b}{2a}, \text{ в інших випадках;}\end{cases}$ $x=\dfrac{\log_a b}{\cos b}$; $r=\begin{cases}2z, \text{ якщо } z>12;\\ 3z, \text{ якщо } z=12;\\ 4z, \text{ якщо } z<12\end{cases}$

2.4 Розгляд основних пунктів виконання завдання 2.2

Зазвичай програмний додаток (HTML-додаток у нашому випадку) розробляється, коли використовувати його збираються багаторазово. Але іноді має сенс скласти програму (сценарій) лише для одноразового використання, коли потрібний розрахунок іншим способом виконавцю зробити складніше. Зрозуміло, що таку програму прагнуть отримати якнайшвидше й так, щоб не треба було її довго налагоджувати. Це можливо, коли вона складається з великої кількості майже однакових фрагментів, які зроблені за допомогою багаторазового копіювання текстовим редактором деякого початкового фрагмента з подальшим невеликим корегуванням цих копій. Цикли в такій програмі не використовуються, щоб не ускладнювати код. Сам же початковий фрагмент настільки простий, що не вимагає налагодження або воно нескладне.

Завдяки вимогам завдання 2.2 моделюється саме така ситуація.

Нехай задано поліном (многочлен) з цілими коефіцієнтами:

$$f(x)=6x^4-x^3-7x^2+x+1 .$$

Згідно з теоремою, що наведена в загальній частині завдання 2.2, для цього полінома число p (дільник вільного члена) може мати два значення: 1 та -1; а число q (дільник коефіцієнта полінома при старшому ступені аргументу) може мати вісім значень: 1 та -1, 2 та -2, 3 та -3, 6 та -6. Таким чином, усі раціональні корені для заданого варіанта полінома треба шукати серед такої множини: 1, -1, 1 /2, -1 /2, 1 /3, -1 /3, 1 /6 та -1 /6.

Оскільки метою складання цього сценарію є розв’язання однієї конкретної задачі (він навіть не матиме програмних можливостей введення даних тому, що всі початкові дані будуть вкладені безпосередньо в код), то розробляти повноцінний контрольний приклад немає сенсу (навіщо потрібен скрипт, коли задача й так буде вже розрахована). Але в правильності обчислень програми переконатися можна: сценарій складатиметься з набору однакових фрагментів (перевірки конкретних чисел як коренів рівняння). Тому, якщо перевірити правильність розрахунку одного фрагмента, то можна з достатньою ймовірністю вважати, що правильно обчислює увесь скрипт. Перевіримо розрахунок полінома тільки для $x = 1$:

$$6x^4 - x^3 - 7x^2 + x + 1 = 6 - 1 - 7 + 1 + 1 = 0,$$

тобто це є один із шуканих коренів полінома.

Схема алгоритму обчислення вибраного варіанта завдання наводиться на рисунку 2.4.

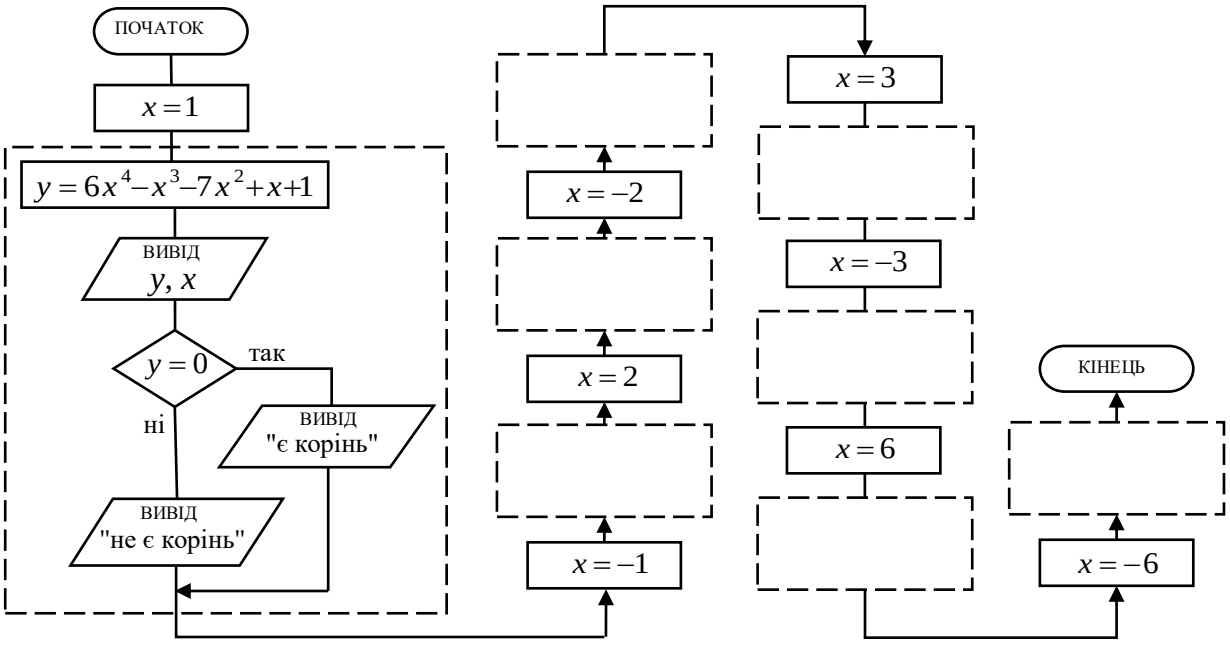


Рисунок 2.4 – Схема алгоритму розв’язання прикладу завдання 2.2

У схемі «порожні» прямокутники з пунктирними лініями є копіями першого від початку «заповненого» прямокутника, тобто

перший фрагмент схеми, що обмежений пунктиром, дублюється багаторазово.

Код прикладу тут не наводитиметься тому, що для всіх варіантів завдання він, практично, однаковий. Але доречною є рекомендація щодо організації виведення результатів. Якщо при кожній потребі виведення даних згідно із схемою застосовувати виклик функції MsgBox, то це призведе до появи черги вікон з повідомленнями, яку треба буде поступово проглядати, закриваючи попереднє вікно, щоб відкрилося наступне. Це дуже незручно. Тому пропонується зробити виведення всіх потрібних даних разом в одному вікні.

Такий спосіб виведення описано у конспекті [2, с. 62–64]. Нагадаємо основну ідею. На початку коду резервується деяка величина (наприклад величина v), до якої заноситься «порожній рядок». Коли в процесі виконання сценарію виникає необхідність виведення якихось даних та коментарів до них, то це все «прив’язується» до величини v за допомогою операції конкатенація. Тобто величина v використовується як накопичувач символьного рядка. Якщо треба у вікні повідомлення візуально подати цей рядок, що накопичується у змінній v, як декілька рядків, то в потрібному місці рядка вставляється іменна стандартна символьна константа vbLF (це робиться також за допомогою конкатенації).

Нехай, наприклад, у змінній y зберігається значення обчислення полінома, а в змінній x – значення аргументу. При цьому з’ясувалося, що поточне значення величини x є коренем полінома, тоді до символьного рядка у змінній v можна додати отримані дані та коментар до них за допомогою такої послідовності операторів:

```
v = v & "f (" & x & ") = " & y & vbLF
v = v & "Значення аргументу є корінь." & vbLF & vbLF  .
```

Подвійне застосування константи vbLF використано для покращення читання повідомлення у вікні, що виводиться у кінці сценарію викликом:

```
MsgBox v, , "Вікно виводу" .
```

Тоді, якщо цей сценарій виконує пошук раціональних коренів полінома, що був наведений як приклад раніше, то у "Вікні виводу" з’явиться, приблизно, таке повідомлення:

"f (1) = 0
Значення аргументу є корінь.

f (-1) = 0
Значення аргументу є корінь.

f (0,5) = 0
Значення аргументу є корінь.

f (-0,5) = -0,75
Значення аргументу НЕ є корінь.

f (0,333333333333333) = 0,592592592592593
Значення аргументу НЕ є корінь.

f (-0,333333333333333) = 0
Значення аргументу є корінь.

f (0,166666666666667) = 0,972222222222222
Значення аргументу НЕ є корінь.

f (-0,166666666666667) = 0,648148148148148
Значення аргументу НЕ є корінь. "

Запис результату першої перевірки в ньому збігається з розрахунком контрольного прикладу.

Завдання 2.2

Мовою VBS треба скласти сценарій, що дає змогу знайти всі раціональні корені деякого полінома *n*-го ступеня з цілими коефіцієнтами. Аналітичний опис полінома обираєте згідно з варіантом із таблиці 2.2.

Для виконання завдання скористайтеся нижченаведеною теоремою.

Теорема. Для того, щоб нескоротний дріб p/q ($q \neq 0$) був коренем рівняння

$$a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0 = 0 \ \ (a_n \neq 0)$$

з цілими коефіцієнтами, необхідно, щоб число p було дільником вільного члена a_0, а число q — дільником старшого коефіцієнта a_n.

Наприклад, якщо рівняння має цілі коефіцієнти, а старший коефіцієнт дорівнює одиниці (тобто $a_n = 1$), то раціональними коренями цього рівняння можуть бути тільки цілі числа, які є дільниками вільного члена a_0.

Метою цього завдання є не тільки отримання виконавцем навичок організації розгалужень у сценаріях VBS, але воно також є прикладом складання сценарію обчислення лише для одного конкретного набору вхідних даних. Такий прийом виконання складного розрахунку можна застосовувати, коли його виконавець розуміє, що іншим способом він отримає результат за довший проміжок часу. Скрипт при такій ситуації, звісно, треба складати так, щоб він був конструктивно простий і «прозорий», який не потребував би складного налагодження. Наприклад, якщо він може складатися із декількох похожих фрагментів, то не треба намагатися організовувати функціонування циклу, а швидко зробити код за допомогою копіювання фрагмента у текстовому редакторі.

Тому після того, як ви отримаєте набір чисел, серед яких треба шукати корені полінома, за годинником зафіксуйте момент початку складання сценарію та момент виведення на екран комп’ютера вікна з повідомленням про результати розрахунку, про тривалість проміжку часу між цими моментами сповістіть викладача.

Якщо виконання лабораторної роботи відбуватиметься у відповідній аудиторії університету, то проведіть змагання між виконавцями завдання. Попросіть викладача бути арбітром у ньому. Виконайте завдання з таблиці 2.2.

Таблиця 2.2

Варіант	Аналітичне задавання полінома
1	x^3-4x^2-x+4
2	$6x^4+7x^3-36x^2-7x+6$
3	x^3+2x^2-x-2
4	$2x^3-4x^2-x-15$
5	$2x^5+5x^4+11x^3+14x^2+11x+5$
6	$x^4+4x^3-2x^2-12x+9$
7	$2x^3-3x^2-3x+2$
8	$3x^3+2x^2-2x+3$
9	$x^4+3x^3-44x^2+15x+25$
10	$2x^3-4x^2-x-15$
11	$6x^4+19x^3-7x^2-26x+12$
12	$12x^5+18x^4-45x^3-45x^2+18x+12$
13	$x^4+2x^3-2x^2-6x+5$
14	$6x^4-7x^3+36x^2+7x-6$
15	$6x^4+5x^3-38x^2+5x+6$

ЛАБОРАТОРНА РОБОТА 3. Розробка графічного інтерфейсу до сценаріїв, складених мовою VBScript, у HTML-додатку

Мета лабораторної роботи – отримання практичних навичок розроблення графічного інтерфейсу до програм, складених алгоритмічною мовою VBScript, із елементів управління та інших компонентів веб-сторінок, заданих уHTML-документі.

Завдання лабораторної роботи

1 Вивчити теоретичний матеріал, який надається в підрозділі 3.1, та матеріали, на які вказують посилання із цього підрозділу.

2 Ознайомитися із завданням 3.1 та із альтернативним (спрощеним) завданням 3.1 і виконати одне з них, використовуючи при розробленні обробників подій скрипти із завдань 1.1 та 2.1 згідно зі своїм варіантом.

3 Скласти звіт про результати виконання лабораторної роботи, до якого включити: записи умов завдання 1.1 та завдання 2.1, код HTML-документа зі сценарієм, покроковий словесний опис процесу розрахунку або відповідний набір «скриншотів».

3.1 Довідка про теоретичні засади, які потрібні при виконанні роботи

Перед виконанням лабораторної роботи її виконавець повинен знати:

теоретичні відомості, що використовувалися при виконанні попередніх лабораторних робіт;

основи мови HTML: поняття: теги як керуючі конструкції мови, побудова тегів, теги парні і непарні, атрибути тегів, значення атрибутів;

структуру HTML-документа, спеціальні теги структури;

способи розміщення коду сценарію в HTML-документі;

теги для створення текстового та графічного вмісту HTML-документа;

компоненти веб-сторінок: пасивні та активні компоненти, елементи управління;

особливості застосування тегу <INPUT>;

одномодульний і процедурний стилі програмування та обробники подій;

властивості змінних при процедурному стилі програмування.

Ці теоретичні матеріали наведені у конспекті лекцій [2, с. 37–54], у розділах: «Процедурний стиль програмування і обробники подій», «Задавання процедур у сценаріях VBScript», «Прив’язка обробників подій до компонентів веб-сторінки HTML-документа».

Програми, складені мовою VBS, найкраще виконувати в браузері Microsoft Internet Explorer (IE) попередньо зробивши їх

частиною HTML-документа. Кожен такий документ описує вміст веб-сторінки, тобто те, що буде показано браузером в одному відкритому його вікні.

Компонентами веб-сторінки можуть бути фрагменти тексту, картинки, віртуальні панелі управління плеєрів для відео та звуку, інші графічні пасивні й активні елементи (поля для введення тексту, віртуальні клавіші, кнопки та ін.) Компонентами веб-сторінки можуть бути також програми, написані скриптовими мовами.

Керуючі конструкції мови HTML називаються тегами (англ. "tag" — іменована мітка) і вставляються безпосередньо в текст документа.

Усі теги укладаються в кутові дужки " <...> ". Відразу після відкриваючої дужки поміщається ключове слово, яке визначає тег. Теги HTML бувають парними і непарними. Непарні теги впливають на весь документ або визначають разовий ефект у місці своєї появи. При використанні парних тегів у документ додаються відкриваючий і закриваючий теги, які впливають на частину документа, укладену між ними.

Закриваючий тег відрізняється від відкриваючого наявністю символу " / " (прямий слеш) перед ключовим словом. Закриття парних тегів виконується так, щоб дотримуватися правил вкладення.

Ефект застосування тегу може змінюватися шляхом додавання атрибутів (властивостей). У парних тегах атрибути додаються тільки до відкриваючого тегу. Атрибути являють собою додаткові ключові слова, які відділяються від ключового слова, що визначає тег, та від інших атрибутів пробілами і розміщуються до символу " > ", що закінчує тег. Спосіб застосування деяких атрибутів вимагає вказівки значення атрибута. Значення атрибута відділяється від ключового слова атрибута символом " = " (знак рівності) і укладається в подвійні або одинарні прямі лапки. Ключі тегів, їхні атрибути зі значеннями можна задавати як великими, так і малими літерами.

Усі HTML-документи мають ту саму структуру, яка визначається фіксованим набором спеціальних тегів структури. HTML-документ завжди починається з тегу <HTML> і закінчується відповідним закриваючим тегом </HTML>.

Усередині документа виділяються два основні розділи: розділ заголовків і тіло документа, які йдуть саме в такому порядку. Розділ заголовків обмежується парним тегом <HEAD> … </HEAD>. Він повинен містити назву документа, яка обмежується парним тегом <TITLE> … </TITLE>. Тіло документа містить зміст самого документа, який обмежується парним тегом <BODY> … </BODY>.

Звісно, положення структурних тегів у документі неважко визначити, навіть якщо вони опущені, тому IE може «читати» документ і без них. Проте при створенні HTML-документа опускати структурні теги небажано.

Код сценарію мовою VBScript повинен розміщуватися всередині парного тегу <SCRIPT>…</SCRIPT>, який у свою чергу може бути вставлений як усередину парного тегу <HEAD>…</HEAD>, так і всередину парного тегу <BODY>…</BODY>. При цьому початковий тег <SCRIPT> має обов’язково оголошувати атрибут "LANGUAGE" зі значенням "VBScript". Рекомендується безпосередньо сам код скрипту вписувати поміж тегами " <!-- " та " --> ". Ці теги в мові HTML обмежують коментарі і, отже, «сповіщають» браузер, що усякий вміст між ними треба ігнорувати при візуалізації документа.

Для організації абзацу тексту на веб-сторінці служить парний тег <P> ... </P> (від англ. "paragraph"). Абзац завжди починається з нового рядка, а абзаци тексту, що йдуть один за одним, розділяються між собою проміжком. Текст в абзаці завжди форматується відповідно до ширини вікна браузера. Парний тег абзацу є також зручним «контейнером» для вдалого розміщення вздовж горизонталі інших компонентів сторінки. Текст абзацу на сторінці подається у два або більше рядків тільки в результаті автоматичного форматування відповідно до розташування та ширини вікна браузера. Щоб у тексті абзацу на веб-сторінці забезпечити примусовий перехід на новий рядок, потрібно використовувати непарний тег
 (від англ. "break" — розрив).

Зазвичай, HTML-документ у своєму складі має як пасивні компоненти веб-сторінок (текст, картинки, фотографії, елементи декору), так і активні компоненти (віртуальні клавіші, кнопки, поля для введення символів тощо). Серед останніх є так звані

елементи управління — компоненти веб-сторінок, які забезпечують обмін даними поміж клієнтським комп'ютером (за допомогою браузера) та веб-сервером, що обслуговує сайт. Однак дані, які надходять від елементів управління, можна і не відсилати на веб-сервер, а залишити в програмному середовищі браузера. Тоді ці елементи управління стають зручними елементами графічного інтерфейсу, який забезпечуватиме зв'язок між сценарієм VBScript та його користувачем.

Елементів управління в HTML досить багато, але вони не всі добре пристосовані для названих цілей. Тому розглядатимуться з них лише два: поле для введення (виведення) текстового рядка та кнопка. Ці елементи задаються на веб-сторінці за допомогою непарного тегу <INPUT>. Цей тег обов'язково повинен мати атрибут "TYPE" (тип), значення якого визначає, який елемент управління відображатиметься на веб-сторінці. Будемо користуватися тільки одним з двох значень цього атрибута: "Button" (кнопка) або "Text" (текстове поле).

Для цих елементів управління в тегу <INPUT> можуть бути наведені ще додаткові атрибути "VALUE" та "SIZE". У випадку, коли елементом управління є текстове поле, значення для "VALUE" задає початковий символьний рядок, що буде первісно мати цей елемент відразу після завантаження. Коли ж елементом управління є кнопка, то значення для "VALUE" задає напис на цій кнопці. Атрибут "SIZE" застосовується тільки для опису текстових полів, його значенням є число — довжина поля, що вимірюється кількістю знакомісць.

Щоб забезпечити зв'язок між компонентами веб-сторінки та сценарієм, деяким тегам треба давати імена (ідентифікатори). Для цього до тегу додається атрибут "ID" зі значенням, яке є, власне, ідентифікатором компоненти веб-сторінки, що задається цим тегом.

Кожен ідентифікатор — це деяка символьна послідовність, яка може складатися з латинських букв, цифр, символів дефіса та підкреслення. Самим лівим знаком у ній повинна бути обов'язково буква. Але ці ідентифікатори треба використовувати й у сценаріях VBScript, тому застосування дефіса в їх значеннях є неприпустимим (цей знак використовується в скриптах VBS для задавання операції віднімання).

Сценарії VBScript можна писати в одномодульному та багатомодульному (процедурному) стилях. Процедура (модуль) у сценарії — це окрема секція коду, що виконує конкретне завдання. При процедурному стилі програмування сценарій складається, зазвичай, із головної процедури та однієї чи декількох допоміжних (підлеглих) процедур. При одномодульному стилі програмування сценарій складається тільки із головної процедури. Головна процедура завжди виконується відразу після завантаження браузером, а допоміжні процедури виконуються за викликом із головної процедури, із іншої допоміжної процедури та навіть із себе самої. Ще є такі допоміжні процедури, які викликаються до виконання при настанні деякої події, що може трапитися в процесі показу веб-сторінки у вікні браузера. Ці процедури називають обробниками подій.

Задавання головної процедури сценарію не потребує введення спеціальних інструкцій до коду: перший її оператор є початком процедури, останній — її кінцем. Усі допоміжні процедури такі спеціальні інструкції мають. Докладніше про це можна дізнатися у конспекті [2, с. 42–46, 48–52].

Достатньо часто розрахункові додатки, розроблені у процедурному стилі програмування, складаються тільки з головного модуля та декількох обробників подій, більш того, головний модуль при цьому може мати лише інструкції-оголошення та навіть зовсім не мати інструкцій (у такому випадку вважається, що місце в HTML-документі поміж тегами <SCRIPT> та </SCRIPT> є головний модуль, а обробники подій додаються до нього).

Прикладом такого коду може бути вміст HTML-додатка, що наводиться нижче.

```
<HTML> <HEAD>
<TITLE>Приклад сценарію тільки із
    одного обробника події</TITLE>
</HEAD> <BODY>
<H1>Отримання квадрата числа</H1>
<P><INPUT TYPE="Text" ID="paramX" SIZE="16">
    : введіть число та двічі клацніть по ньому</P>
<SCRIPT LANGUAGE="VBScript"> <!--
```

```
SUB paramX_ondblclick
        x = 0 + paramX.Value
        paramX.Value = x * x
END SUB
--> </SCRIPT> </BODY> </HTML>
```

У цьому прикладі демонструється як задається обробник події (його код обмежується рядком з ключем SUB та рядком END SUB), як пов’язується він з елементом управління веб-сторінки та подією (за допомогою свого складеного ім’я — це ім’я складається із значення атрибута "ID" елемента управління, з яким має бути пов’язаний обробник події, та ідентифікатора події, що поєднуються символом підкреслення "_"). Також у прикладі показаний спосіб передачі даних від елемента управління "текстове поле" до змінної, що належить обробнику події, та зворотно — від змінної обробника події до елемента управління "текстове поле" (елемент управління розглядається як об’єкт, що задається значенням атрибута "ID", який має властивість "VALUE", для визначення якої застосовується спеціальний оператор-крапка, що поєднує разом значення ідентифікатора елемента управління та назву його властивості для застосування як посилання у складі оператора присвоювання). Детальніше про це подано у конспекті [2, с. 47–50].

Процедурний стиль програмування надає змінним, що застосовуються у сценарії, додаткові властивості: області видимості і час життя, завдяки яким можна встановлювати простий обмін даними між процедурами сценарію (докладно дивись [2, с. 13–15]).

3.2 Розгляд основних пунктів виконання завдання 3.1

Виконавцю цього завдання треба прагнути так задати веб-сторінку, щоб вона стала зручним інтерфейсом при керуванні обчисленнями за сценаріями, що були вже розроблені ним згідно з варіантом із завдань 1.1 та 2.1 попередніх лабораторних робіт.

Йому також треба намагатися розмістити компоненти цієї веб-сторінки, написи до них та коментарі так, щоб сторінка візуально мала гарний вигляд. Для цього можна використовувати

теги абзацу як контейнери для розміщення елементів управління вздовж горизонталі вікна браузера. До початкового тегу <P> іноді треба додати атрибут "ALIGN" (вирівняти), який може задавати значення "Center" (вирівнювання по центру), "Right" (вирівнювання по правому краю) та "Justify" (вирівнювання по ширині).

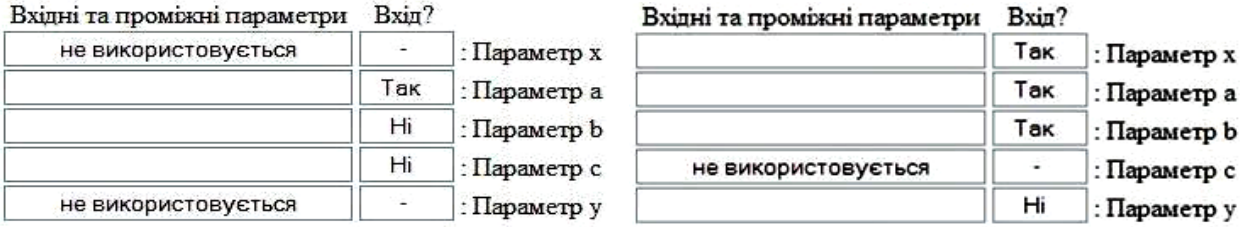
Написи до текстових полів та кнопок робляться також у тегах абзаців. Якщо треба їх вирівнювати відносно елементів управління, то рекомендується здійснювати це вручну шляхом додавання до рядка абзацу пробілів. Але для цього треба використовувати спеціальні символи «нерозривного пробілу», які можна вставити до рядка за допомогою введення до нього символьної послідовності   , яку треба повторювати стільки разів, скільки треба ввести пробілів. Застосовувати для цього послідовності із звичайних пробілів неможливо, оскільки звичайний пробіл у HTML-документі використовується як нейтральний роздільник між всякими його елементами, тому будь-яка послідовність з двох і більше звичайних пробілів сприймається браузером при візуалізації веб-сторінки завжди як один звичайний пробіл, навіть якщо ця послідовність міститься в тексті абзацу.

Для загальних заголовків рекомендується застосовувати парний тег <H1> ... </H1>, який діє як теги абзацу, але свій вміст задає найбільшим жирним шрифтом з усіх тих, які використовуються браузером для виведення заголовків. Для задавання заголовків з більш дрібними жирними шрифтами використовуються теги з ключами "H2", "H3" і так до "H6". Останній із ключів визначає найдрібніший для заголовків шрифт.

До складу сценарію повинно входити чотири обробники подій: два обробники події "onclick", що безпосередньо виконують обчислення завдань 1.1 та 2.1, та два обробники подій "onclick" і "ondblclick", які підготовляють загальний блок текстових полів для введення та відображення значень вхідних та проміжних параметрів разом із додатковими текстовими полями для зазначення їх статусів перед виконанням обчислень. Останні два обробники подій «прив’язуються» до однієї кнопки "Підготувати поля!", але при різних подіях. Однак треба мати на увазі, що подія "ondblclick" завжди трапляється після події

"onclick", тобто подія "onclick" обов’язково передує їй. Тому результати виконання обробника події "onclick" ніяк не повинні відображатися на результатах виконання обробника події "ondblclick"— це є обов’язкова умова складання останнього. Через це у ситуації, що визначається цим завданням, запропоноване в ньому об’єднання «запусків» двох обробників події спільною кнопкою є єдино можливим.

Якщо згідно із завданням 3.1 розробити графічний інтерфейс для обчислень, що були розглянуті як приклади при виконанні завдань 1.1 та 2.1 у попередніх лабораторних роботах, то загальний блок текстових полів та додаткові поля для зазначення статусів після того, як відпрацюють відповідні обробники подій, повинні мати, приблизно, такий вигляд, як на рисунку 3.1.



а) б)

Рисунок 3.1 – Загальний блок текстових полів з додатковими полями для зазначення статусів, що підготовлені до введення даних для обчислення прикладу завдання 1.1 (а) і що підготовлені до введення даних для обчислення прикладу завдання 2.1 (б)

Щодо загального вигляду всього графічного інтерфейсу, то при дотриманні повного набору умов він може мати такий вигляд у вікні браузера, як показано на рисунку 3.2. Дані, що наведені у текстових полях на рисунку, відповідають моменту закінчення розрахунку завдання 1.1 за наданим контрольним прикладом.

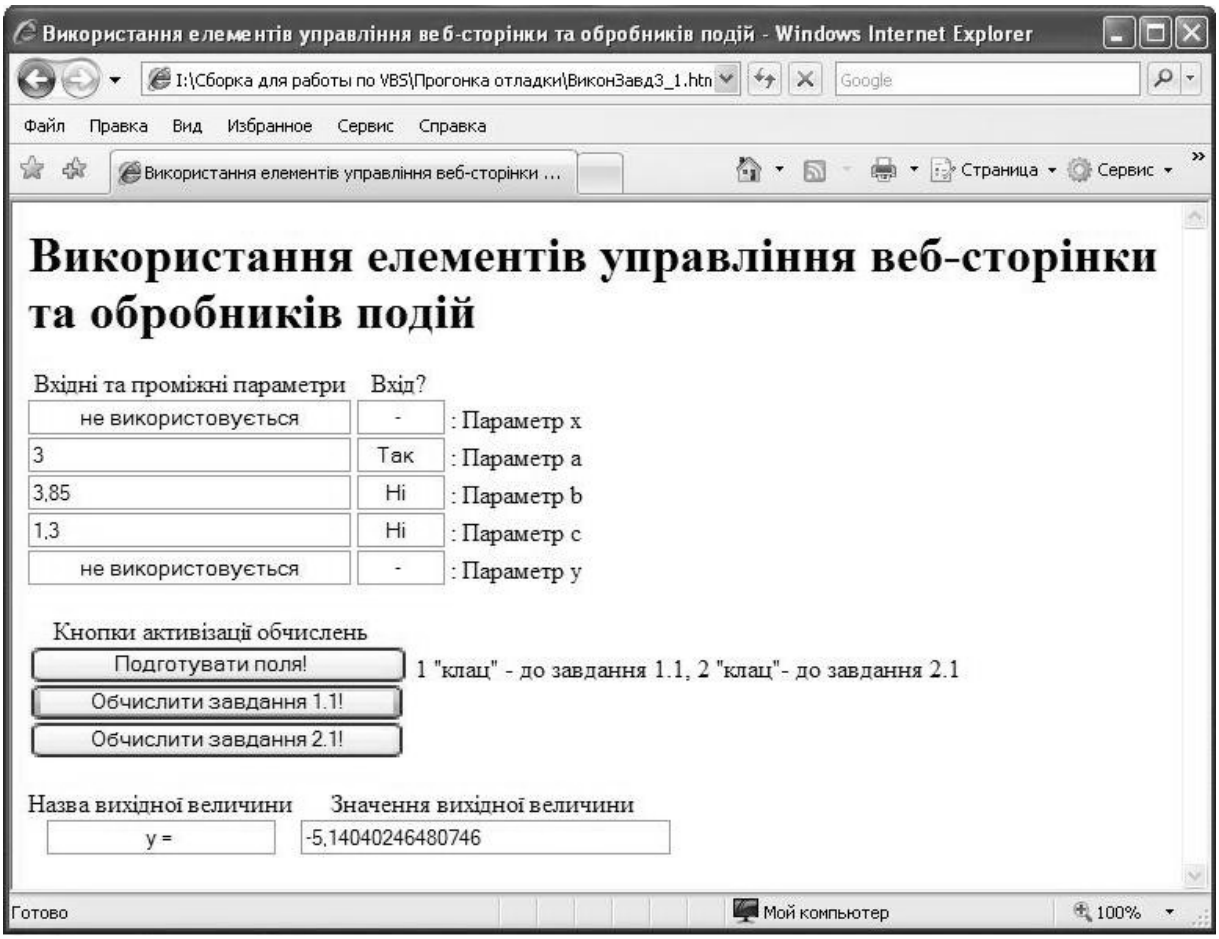


Рисунок 3.2 – Приклад графічного інтерфейсу HTML-додатка, складеного згідно із завданням 3.1

Завдання 3.1

Застосовуючи елементи управління та інші компоненти веб-сторінки, розробити графічний інтерфейс для спільного використання сценаріїв до завдань 1.1 та 2.1 відповідно до свого варіанта. Результатом виконання завдання повинен стати HTML-документ, який дає змогу проводити обчислення як за сценарієм виконання завдання 1.1, так і за сценарієм виконання завдання 2.1, і графічний інтерфейс (веб-сторінка) якого має такі властивості та відповідає вимогам:

1) інтерфейс має загальний блок текстових полів для введення та відображення значень вхідних та проміжних параметрів. Якщо за описами завдань 1.1 та 2.1 є вхідні та проміжні параметри, назви яких збігаються, то в загальному блоці текстових полів для обслуговування таких величин

повинно бути лише одне поле, що буде спільне у використанні. Біля кожного поля з цього блоку також потрібно вказати ім’я відповідного параметра;

2) у інтерфейсі поруч з кожним текстовим полем із загального блоку текстових полів повинно бути розміщено невелике (довжина 5-6 знакомісць) текстове поле, куди заносяться в результаті підготовки текстових полів до того чи іншого завдання (опис цієї дії дається нижче) рядки: або "Так", або "Ні", або "-" ("Так" – відповідний параметр є вхідним, "Ні" – параметр є проміжним, "-" – параметр в обчисленні вибраного завдання не використовується);

3) після дії підготовки текстових полів до виконання завдання 1.1 або завдання 2.1 у текстові поля для значень параметрів, що не задіяні в обчисленнях завдання, треба, щоб було внесено рядок "не використовується";

4) загальний блок текстових полів для введення та відображення значень вхідних та проміжних параметрів повинен мати відповідний напис-заголовок: "Вхідні та проміжні параметри", невеличкі текстові поля, що показують статус параметрів у обчисленні, теж повинні мати свій загальний заголовок: "Вхід?";

5) інтерфейс має блок текстових полів для виведення результату обчислення сценарію, що відбувся. Цей блок складається із двох текстових полів, що мають написи: "Назва вихідної величини" та "Значення вихідної величини";

6) інтерфейс має блок кнопок активізації обчислень, до якого належать: кнопка "Підготувати поля!", кнопка "Обчислити завдання 1.1!" та кнопка "Обчислити завдання 2.1!". Цьому блоку треба дати заголовок: "Кнопки активізації обчислень". Кнопка "Підготувати поля!" має біля себе коментар до використання, у якому вказано, що при дії "одного клацання миші" по цій кнопці відбувається підготування текстових полів до обчислення за сценарієм виконання завдання 1.1, а при дії "подвійного клацання миші" по кнопці — підготування текстових полів до обчислення за сценарієм виконання завдання 2.1;

7) дії при активізації кнопки "Підготувати поля!" повинні бути такі: (при дії "одне клацання миші") у загальному блоці текстових полів у поля, що пов’язані з параметрами, імена яких

відсутні в умові завдання 1.1, вноситься символьний набір з повідомленням "не використовується", інші поля блоку "очищаються"; до полів, що показують статус параметрів у обчисленні, вносяться символьні константи "Так", "Ні" та "-" згідно з умовою завдання 1.1; текстові поля, що належать до блоку полів для виведення результату "очищаються"; (при дії "подвійне клацання миші") відбуваються ті ж самі дії, але відповідно до особливостей умови завдання 2.1;

8) дії при активізації кнопки "Обчислити завдання 1.1!" повинні бути такі: викликається й виконується сценарій обчислення завдання 1.1; текстові поля загального блоку полів для введення та відображення значень вхідних та проміжних параметрів отримають відповідні значення для проміжних параметрів (ті текстові поля, для яких поля, що показують статус параметрів, мають значення "Ні"); текстові поля, що належать до блоку полів для виведення результату, теж отримують свої значення: поле для назви вихідної величини отримає потрібне ім’я вихідної величини, поле для значення — отримає числове значення, що було обчислене під час виконання сценарію;

9) дії при активізації кнопки "Обчислити завдання 2.1!" повинні бути такі ж самі, що й для пункту 8, але відповідно до особливостей умови завдання 2.1 та за результатами обчислення сценарію, який виконує це завдання.

Послідовність дій користувача та зміни зовнішнього вигляду інтерфейсу при застосуванні цього HTML-додатка повинні бути такі:

1) HTML-документ завантажується в IE відомим способом. У вікні браузера з’являється відображення веб-сторінки, що є інтерфейсом;

2) після відкриття декількох діалогових вікон, у яких браузер застерігає користувача про можливу небезпеку застосування HTML-додатка (користувач повинен дозволити всі дії) інтерфейс додатка стає активним (текстові поля стають доступними для внесення до них символів). При цьому всі текстові поля є порожні;

3) користувач обирає за яким завданням треба робити обчислення. Для цього він активізує кнопку "Підготувати поля!" шляхом "одиночного клацання миші" або "подвійного клацання

миші" (для виконання обчислень за завданням 2.1). У результаті текстові поля вхідних та проміжних параметрів очищаються, якщо туди раніше було щось внесено. Але в деяких з них може виникнути напис "не використовується". У текстових полях, що показують статус параметрів при обчисленнях, з'являються написи "Так", "Ні" чи "-" згідно з умовами завдання, що обрано для розрахунку. Текстові поля для виведення результату залишаються порожні або очищаються, якщо там щось було;

4) користувач вносить до полів для введення та відображення значень вхідних і проміжних параметрів, які у відповідних полях статусу помічені написом "Так", потрібні числові вхідні дані та "клацає мишею" по одній з двох кнопок "Обчислити…". У результаті всі поля, що повинні мати числові дані, отримають необхідні дані, а поле для назви вихідної величини отримає її ім'я.

Зауваження. Не можна підготовляти поля під виконання одного завдання, а обчислювати інше. Це спричинить конфлікт даних при обчисленнях.

3.3 Розгляд основних пунктів виконання альтернативного завдання 3.1

Застосування елементів управління веб-сторінок для «спілкування» користувача із обчислювальним процесом сценарію є, дійсно, зручний спосіб обміну даними, який не гальмує роботу браузера і дозволяє користувачу ознайомитися з результатами розрахунків коли це йому потрібно й навіть роздрукувати їх, скориставшись відповідною послугою браузера.

Звісно, щоб інтерфейс, який задається на веб-сторінці, був зрозумілий користувачеві, мав візуально стильний вигляд тощо, необхідно розробляти для цього допоміжне облаштування, наприклад, застосовувати спеціальні обробники подій, які в потрібний момент змінюють функції елементів управління та коментарі до них (таким є інтерфейс, що треба розробити згідно із завданням 3.1). Але можна (і це теж зручно) використовувати кожен елемент інтерфейсу лише для однієї своєї функції. Розробити такий графічний інтерфейс пропонується в альтернативному (спрощеному) завданні 3.1.

Нижче наводиться код HTML-документа прикладу виконання такого завдання. Умови обчислень для двох обробників подій, що використовуються у цьому прикладі, взяті із прикладів виконання завдань 1.1 та 2.1 попередніх лабораторних робіт. Для перевірки правильності обчислень можна також узяти ті ж самі контрольні приклади, що застосовувалися при виконанні завдань 1.1 та 2.1.

Код HTML-документа прикладу виконання альтернативного завдання 3.1:

```
<HTML> <HEAD>
<TITLE>Використання елементів управління веб-сторінки
та обробників подій </TITLE>
</HEAD> <BODY>
<H1>Використання елементів управління веб-сторінки та
обробників подій </H1>
<P> Вхідні та проміжні параметри для
завдання 1.1<BR>
<INPUT TYPE="TEXT" ID="ParA1" SIZE="30"> 
: Параметр a (вхідний) <BR>
    <INPUT TYPE="TEXT" ID="ParB1" SIZE="30"> 
: Параметр b <BR>
    <INPUT TYPE="TEXT" ID="ParC1" SIZE="30"> 
: Параметр c </P>
<P> Вхідні та проміжні параметри для
завдання 2.1<BR>
    <INPUT TYPE="TEXT" ID="ParA2" SIZE="30"> 
: Параметр a (вхідний) <BR>
    <INPUT TYPE="TEXT" ID="ParB2" SIZE="30"> 
: Параметр b (вхідний) <BR>
    <INPUT TYPE="TEXT" ID="ParX2" SIZE="30"> 
: Параметр x (вхідний) <BR>
    <INPUT TYPE="TEXT" ID="ParY2" SIZE="30"> 
: Параметр y</P>
<P> <INPUT TYPE="BUTTON" ID="Knop11"
VALUE="Обчислити завдання 1.1!"> <BR>
     <INPUT TYPE="BUTTON" ID="Knop21"
VALUE="Обчислити завдання 2.1!"> </P>
<P>Назва вихідної величини     
```

Продовження коду

```
        &amp;#160;&amp;#160;Значення вихідної величини <BR>
        &amp;#160;&amp;#160;&amp;#160;
        <INPUT TYPE="TEXT" ID="Im_vel" SIZE="20">
        &amp;#160;&amp;#160;&amp;#160;&amp;#160
        <INPUT TYPE="TEXT" ID="Znac" SIZE="35"></P>
<SCRIPT LANGUAGE="VBScript"><!--
SUB Knop11_onclick
        DIM a, b, c, y
        a = 0 + ParA1.Value
        c = 1.3 : b = ( a ^ 2 - c) / 2
        y = a / b + (Sin( a ) ^ 2 - 8) / Log( b )
        Im_vel.Value = "                y = "
        Znac.Value = y
        ParB1.Value = b
        ParC1.Value = c
        END SUB
SUB Knop21_onclick
        DIM a, b, x, z, y
        x = 0 + ParX2.Value
        a = 0 + ParA2.Value
        b = 0 + ParB2.Value
        IF x > 0 THEN
            y = (a -3 ) ^ 2
        ELSEIF x = 0 THEN
            y = 2 * a
        ELSE
            y = 5
        END IF
        SELECT CASE True
            CASE a > b And b >  y : z = a * b - y ^ 2
            CASE b > a Or y > b  : z = 2 * b
            CASE ELSE : z = 3 * a
        END SELECT
        ParY2.Value = y
        Im_vel.Value = "                z = "
        Znac.Value = z
        END SUB
--></SCRIPT> </BODY> </HTML>
```

Зауваження щодо деяких особливостей коду прикладу.

Для кращого візуального сприйняття інтерфейсу було виконано вирівнювання написів до текстових полів та задані проміжки серед останніх. Це було зроблено введенням у рядки абзаців символів "нерозривного пробілу" (дивись відповідний опис у підрозділі 3.2).

Текстове поле, що має ідентифікатор "Im_vel", використовується для запису у нього імені вихідної змінної, яка отримує результат обчислення за умовою завдання. Це здійснюється внесенням у це поле потрібного символьного рядка. Оскільки ім’я вихідної змінної, як правило, дуже коротке (один символ), а довжина текстового поля є достатньо великою, то для центрування напису в полі до символьного рядка додають потрібну кількість пробілів.

3.4 Альтернативне (спрощене) завдання 3.1

Застосовуючи елементи управління та інші компоненти веб-сторінки, розробити графічний інтерфейс для спільного використання сценаріїв до завдань 1.1 та 2.1 відповідно до свого варіанта. Результатом виконання завдання повинен стати HTML-документ, який дає змогу проводити обчислення як за сценарієм виконання завдання 1.1, так і за сценарієм виконання завдання 2.1, і графічний інтерфейс (веб-сторінка) якого має такі властивості та відповідає вимогам:

1) інтерфейс має два блоки текстових полів для введення та відображення значень вхідних та проміжних параметрів із завдань: один блок — для параметрів завдання 1.1, другий блок — для параметрів завдання 2.1. Усі текстові поля повинні мати підписи, у яких вказано, з яким параметром пов’язане те чи інше поле. У підписах повинні бути також зазначені окремо вхідні параметри;

2) інтерфейс має дві кнопки: кнопку з написом "Обчислити завдання 1.1!" (для активізації сценарію розрахунку завдання 1.1) та кнопку з написом "Обчислити завдання 2.1!" (для активізації сценарію розрахунку завдання 2.1). Обидві кнопки реагують на дію "одиночне клацання миші" (onclick);

3) інтерфейс має два текстових поля для виведення результатів розрахунків за обома сценаріями: у перше текстове поле виводиться ім’я вихідної величини за умовою завдання, у друге текстове поле — її числове значення, що було знайдене в процесі обчислень за сценарієм. Обидва сценарії використовують для виведення даних ці текстові поля спільно.

ЛАБОРАТОРНА РОБОТА 4. Підготовка і розв’язання на ПЕОМ задач за простими циклічними алгоритмами

Мета лабораторної роботи – отримання практичних навичок підготовки, налагодження та виконання програм з простими циклами, складених алгоритмічною мовою VBScript.

4.1 Завдання лабораторної роботи

1 Вивчити теоретичний матеріал, який надається в підрозділі 4.2, та матеріали, на які вказують посилання із цього підрозділу.

2 Виконати відповідно до наведеної умови завдання 4.1 за своїм варіантом. При виконанні завдання для організації циклів у коді сценарію використовується оператор циклу за параметром For...Next, який добре пристосований для організації арифметичних (регулярних) обчислювальних циклів. Введення даних до сценарію треба забезпечувати елементами управління мови HTML. Для виведення даних за умовою завдання пропонується застосовувати два способи: за допомогою функції MsgBox та через створення додаткової веб-сторінки при застосуванні методу Write об’єкта Document. Для кожного з обох варіантів оформлюється відповідно свій сценарій та HTML-документ.

3 Виконати відповідно до наведеної умови завдання 4.2 за своїм варіантом. Для організації ітераційних обчислень застосовувати оператор організації циклів за умовою Do...Loop. Вхідні дані вводяться до середовища обчислень із клавіатури шляхом застосування відповідних елементів управління HTML, спосіб виведення отриманих результатів розрахунку обрати за своїм бажанням.

4 Скласти звіт про результати виконання лабораторної роботи згідно з вимогами до змісту звіту, які наведено у розділі 1.

4.2 Довідка про теоретичні засади, які потрібні при виконанні роботи

Перед виконанням лабораторної роботи її виконавець повинен знати:

теоретичні відомості, що використовувалися при виконанні попередніх лабораторних робіт;

способи організації циклічних обчислень: організацію циклу за параметром, організацію циклу за умовою, організацію циклу повного перебору групи (множини);

організацію циклів за параметром засобами мови VBScript (формат та особливості роботи з оператором For...Next);

правила задавання кроку параметра циклу в операторі For...Next;

організацію циклів за умовою засобами мови VBScript (формат та особливості роботи з оператором Do...Loop, п’ять форм його застосування);

можливості та спосіб застосування методу Write об’єкта Document при створенні нової веб-сторінки у вікні браузера.

Ці теоретичні матеріали наведені у конспекті лекцій [2, с. 60–70], у розділі «Організація циклічних обчислень у сценаріях VBScript».

Оператор організації циклу дає змогу повторити виконання деякої кінцевої послідовності інших операторів (цю послідовність зазвичай називають тілом циклу) кілька разів відповідно до заданих умов повтору.

У мові VBScript застосовується три способи організації циклів: організація циклу за параметром, організація циклу за умовою, організація циклу повного перебору групи (множини). Але при виконанні завдань цієї лабораторної роботи знадобляться лише два перших.

Спосіб організації циклу за параметром завжди пов’язаний зі зміною значення деякої числової змінної (однієї змінної, а не кількох), яка змінює своє значення на деяку величину (цю величину називають кроком) відразу після виконання поточної

"петлі" тіла циклу і перед початком виконання наступної "петлі". Таку змінну називають змінною циклу або параметром циклу.

Для організації циклу за параметром у мові VBScript застосовують спеціальний оператор, який часто називають за його ключами: оператор циклу For...Next. Розглянемо використання цього оператора на прикладі розв'язання такої задачі. Треба знайти корінь рівняння

$$\frac{\cos x}{2x+1} = 0$$

у діапазоні дослідження *x* від 0 до 2 з похибкою, не більшою ніж 0,1. Функція лівої частини рівняння на відрізку [0; 2] є неперервна та монотонна, тому, якщо на межах цього відрізка (або на його якійсь частині) зазначена функція є різного знака, то згідно з другою теоремою Вейєрштрасса рівняння на відрізку (або його частині) має єдиний корінь. Тоді, якщо цей відрізок (або його частина) матиме довжину, не більшу ніж 0,1 , то задача буде розв'язана, оскільки кожна точка такого відрізка (частини відрізка) може розглядатися як корінь рівняння, що знайдений із заданою похибкою.

Процедуру пошуку можна задати таким фрагментом коду, у якому головним є оператор For...Next:

```
a = 1
c = 0
FOR x = 0.1 TO 2 STEP 0.1
    b = Cos( x ) / (2 * x + 1)
    IF a * b <= 0 THEN c = 1 : EXIT FOR
    a = b
NEXT
IF c = 1 THEN
    MsgBox "За корінь можна взяти " & (2 * x - 0.1) / 2 & " ."
ELSE
    MsgBox "У заданому діапазоні корінь не знайдено."
END IF
```

Оператор For...Next забезпечує розбиття діапазону пошуку кореня на ділянки довжиною 0,1 та перевірку на кожній з них виконання критерію згідно з теоремою. Змінна *x* у коді

розглядається як параметр циклу. Вона змінює своє значення від 0,1 (на крок більше, ніж початок діапазону дослідження, тобто більше від нуля) й до 2 (кінця діапазону дослідження). Після виконання кожної "петлі" циклу (тих дій, що у фрагменті коду записані нижче рядка з ключем "FOR" та вище рядка з ключем "NEXT") значення параметра циклу змінюється на величину кроку (на 0,1 у цьому прикладі), тобто додається до значення змінної *x* значення кроку. Повторення "петлі" циклу зі зміною значення параметра циклу буде відбуватися, доки значення параметра циклу не перевищить значення виразу, що у рядку з ключем "FOR" розташований правіше ключа "TO".

Запис кроку зміни параметра циклу, що в початковому рядку оператора For...Next, можна пропускати разом із ключем "STEP", якщо його значення є одиниця. В усіх інших випадках крок треба вказувати обов’язково. Якщо треба терміново зупинити циклічні дії оператора For...Next, то застосовують для цього оператор екстреного виходу із циклу Exit For.

Крок зміни параметра циклу може бути як додатною, так і від’ємною величиною. Якщо крок додатний, то початкове значення параметра циклу має бути меншим від його граничного значення, при від’ємному кроці — навпаки. Завдання нульового кроку циклу призводить до "зациклення" сценарію, тобто він не може завершитися, циклічне обчислення в ньому буде повторюватися і повторюватися.

Більш детально про використання оператора For...Next можна прочитати у [2, с. 60–63].

Спосіб організації циклу за умовою полягає в задаванні тіла циклу і деякої керуючої умови, яка може виконуватися або не виконуватися в певний момент процесу обчислень. Акт перевірки умови розглядається як компонент тіла циклу. Якщо умова виконується (або не виконується), приймається рішення про виконання іншої частини тіла циклу або про виконання наступного витка циклу. Останнє залежить від того, на початку або в кінці тіла циклу перебуває дія з перевірки умови.

У VBS є два оператори організації циклів за умовою. Найбільш "універсальним" з них є оператор циклу Do...Loop. Цей оператор може організовувати цикли як з передумовою, так і з післяумовою, при цьому, якщо застосовувати різні допоміжні

ключі, то можна організовувати повторювання тіла циклу при виконанні або при невиконанні керуючої умови. Також оператор Do...Loop може використовуватися для створення "зациклення" сценарію. Тоді обірвати повторення дій можна тільки застосувавши оператор екстреного виходу Exit Do. Зазвичай оператор Exit Do використовується спільно з оператором If...Then...Else.

Таким чином, за допомогою оператора організації циклів за умовою Do...Loop можна створювати циклічні обчислення за п’ятьома формами.

Розглянемо використання цього оператора на прикладі розв’язання тієї ж задачі, що була застосована для розгляду оператора For...Next. Нижче наводиться фрагмент коду із оператором Do...Loop, який використовується для створення циклу з післяумовою при повтореннях тіла циклу, якщо керуюча умова виконується.

```
b = 1 : x = 0.1
DO
    a = b
    x = x + 0.1
    b = Cos( x ) / (2 * x + 1)
LOOP WHILE  a * b > 0 And x <= 2
IF x <= 2 THEN
    MsgBox "За корінь можна взяти " & (2 * x - 0.1) / 2 & " ."
ELSE
    MsgBox "У заданому діапазоні корінь не знайдено."
END IF
```

Тут також відбувається послідовна перевірка ділянок довжиною 0,1 діапазону дослідження *x*. Припинення повторів тіла циклу трапиться, коли знайдеться ділянка діапазону дослідження, на межах якої функція лівої частини рівняння змінює знак або всі ділянки діапазону будуть перебрані. Про те, що в коді прикладу тіло циклу повторюється, коли виконується керуюча умова, вказує застосування там допоміжного ключа "WHILE" (якщо було інакше, то там би використовувався ключ "UNTIL"). А про те, що цикл, який заданий у коді прикладу, є циклом з післяумовою, свідчить місцезнаходження допоміжного

ключа. Він розташований у рядку, де є ключ "LOOP", інакше він розміщувався б поруч з ключем "DO".

Більш докладно про використання оператора Do...Loop говориться у [2, с. 67–69].

4.3 Розгляд основних пунктів виконання завдання 4.1

Головна мета цього завдання — надати виконавцеві можливість при складанні сценарію VBScript застосувати оператор організації циклу за параметром FOR…NEXT для отримання ним початкових навичок у цьому. Тому розв'язання задачі табуляції функціональної залежності, що лежить в основі усіх варіантів завдання, є вдалим вибором для досягнення цієї мети. Параметром циклу там може слугувати аргумент функції, а межі діапазону його дослідження використовуються як початкове та кінцеве значення зміни параметра циклу. Кроком зміни параметра циклу вибирається відстань між сусідніми значеннями аргументу, яка при дослідженні скрізь вважається однаковою.

У всіх варіантах завдання, крім розрахунку значення функції, у кожному відліку аргументу ще здійснюється деякий простий статистичний аналіз різних характеристик функції, який також не можна виконати без застосування арифметичного циклу.

Прикладом варіанта завдання для розгляду окремих його пунктів виберемо такі умови. Треба скласти сценарій мовою VBS табуляції функції:

$$y = \cos bx + x^3 - ax^2 + 2,$$

діапазон зміни аргументу $x1 \le x \le x2$, крок зміни x вводиться користувачем через компоненти інтерфейсу веб-сторінки. Додатково сценарій повинен виконати нижченаведене дослідження:

Визначити кількість значень функції, що були отримані в процесі табуляції, для яких виконується нерівність $f(x_i) > 1/x_i$, та розрахувати їх відсоткове співвідношення до загальної кількості.

Загальна схема алгоритму, за якою треба розробляти сценарій, наведена на рисунку 4.1.

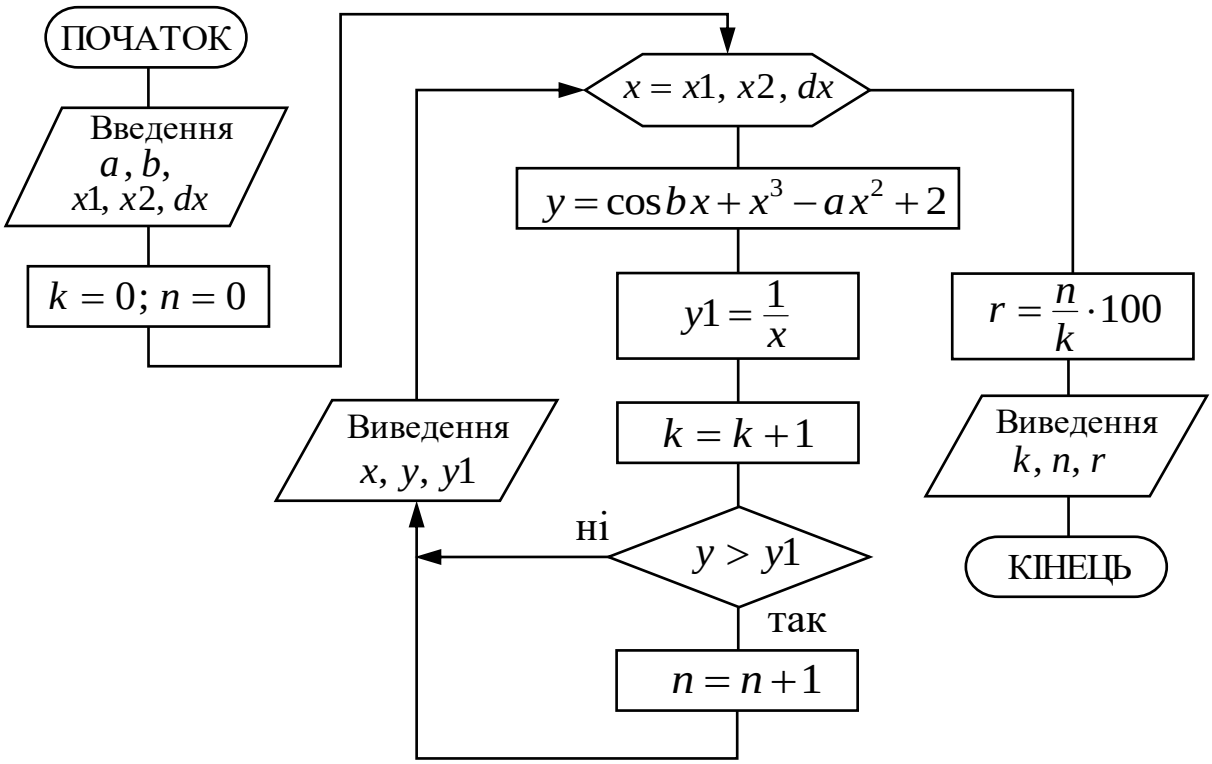


Рисунок 4.1 – Схема алгоритму розв’язання прикладу завдання 4.1

Зі схеми бачимо, що користувачеві треба ввести до обчислювального середовища сценарію п’ять значень вхідних величин: значення параметрів *a* та *b*, що потрібні для обчислення функції, яку досліджують, значення величин *x*1 та *x*2, щоб установити межі розгляду зміни аргументу функції та величину *dx*, що задає крок зміни аргументу. Останні три величини застосовуються далі для задавання роботи змінної арифметичного циклу.

Цей цикл задає всі основні обчислення алгоритму: у ньому безпосередньо відбувається процес табуляції заданої функції, а також здійснюється отримання, перевірка та накопичення даних для розрахунку після закінчення циклічних дій потрібної статистичної характеристики. Для накопичення даних застосовуються змінні *n* та *k*, що використовуються як лічильники. Величина *k* збільшує своє значення на одиницю, коли обчислюється чергове значення функції, а величина *n* підраховує, скільки значень функції виявилося більшими за

значення оберненої величини аргументу, яке з кожною новою "петлею" циклу обчислюється й зберігається у змінній *y*1.

Кожна "петля" циклу закінчується виведенням значення аргументу та відповідного значення функції, а також значення оберненої величини аргументу для можливості порівняння.

Контрольні приклади для перевірки правильності обчислень сценарію для цього завдання можна обмежити розрахунком функції в одній чи у двох точках із набору значень аргументу. Також треба підібрати значення величин *x*1, *x*2 та *dx* так, щоб результат розрахунку "статистики" легко було перевірити вручну.

Код HTML-документа зі сценарієм наведеного прикладу може бути таким:

```
<HTML>
<HEAD>
<TITLE> Табуляція та дослідженні функції </TITLE>
</HEAD>
<BODY ID="StartS">
  <P><INPUT TYPE="Text" NAME="ImyaA" ID="paramA"
SIZE="7">: параметр A<BR>
 <INPUT TYPE="Text" ID="paramB" SIZE="7"> :  параметр B<BR>
 <INPUT TYPE="Text" ID="paramDx" SIZE="7"> : крок зміни
    X<BR>
 <INPUT TYPE="Text" ID="paramX1" SIZE="7"> : почат. знач.<BR>
 <INPUT TYPE="Text" ID="paramX2" SIZE="7"> : кінц. знач.</P>
<SCRIPT LANGUAGE="VBScript"> <!--
SUB StartS_ondblclick
        a = 0 + paramA.Value
        b = 0 + paramB.Value
        dx = 0 + paramDx.Value
        x1 = 0 + paramX1.Value
        x2 = 0 + paramX2.Value
        xv = "x = "
        yv = " y = "
        s = " 1/x = "
        v = "<BODY><P>"
        Document.Write _
        "<TITLE> Вікно виводу результатів </TITLE>"
        k = 0 : n = 0
```

Продовження коду

```
        FOR x = x1 TO x2 STEP dx
           y = Cos( b * x ) + x ^ 3 - a * x ^ 2 + 2
           k = k + 1
           y1 = 1 / x
           IF y > y1 THEN
              y1 = "" &  y1 & " +"
              n = n + 1
           END IF
           v = v & xv & x & yv & y & s & y1 &"<BR>"
         NEXT
       v = v & "Загальна кількість отриманих значень функції: " & k &
       "<BR>"
       v = v & "Кількість значень функції, при яких виконується
умова:"
       & n & "<BR>"
        r = n / k * 100
         v = v & "Це складає " & r & " відсотків від загальної
кількості обчислених значень функції."
         v = v & "</P></BODY>"
         Document.Write v
END SUB
--> </SCRIPT> </BODY> </HTML>
```

Це є варіант коду сценарію, при виконанні якого виведення даних здійснюється в окрему веб-сторінку за допомогою застосування методу Write об’єкта Document. Щоб виведення було коректним, спершу у деякій одній змінній (змінна v в цьому коді) шляхом використання конкатенації формується символьна послідовність, яка за вмістом є фрагментом коду веб-сторінки, куди розміщуються записи результатів розрахунків. Для того, щоб записи розміщалися у декілька рядків, у цю послідовність, де потрібно, вставляються теги
. Далі ця послідовність розгортається у нову веб-сторінку через застосування методу Write.

Якщо виведення у сценарії потрібно зробити за допомогою функції MsgBox, то можна використати той самий прийом із формуванням символьної послідовності в одній змінній, яку потім виводять у вікні на екрані монітора за допомогою лише одного виклику функції MsgBox. Звісно, що така послідовність

не повинна вже вміщувати у собі теги HTML, а замість тегу
, де потрібний розрив рядка, вставляється стандартна константа vbLF.

Завдання 4.1

Мовою VBS треба розробити сценарій процедури табуляції деякої функціональної залежності $y = f(x)$ у заданому діапазоні зміни її аргументу $x1 \le x \le x2$ із заданим кроком зміни Δx та додаткового її дослідження. Описи додаткових досліджень та пов'язані з ними літерні ідентифікатори наведені у списку нижче. Введення даних до сценарію забезпечується елементами управління мови HTML. Виведення даних треба розробити двома способами: 1) за допомогою одного загального вікна виведення функції MsgBox; 2) створенням додаткової веб-сторінки за допомогою методу Write об'єкта Document. Для цього оформлюються відповідно два HTML-документи.

При перевірці працездатності та правильності роботи сценарію виконавець завдання вибирає значення всіх потрібних параметрів самостійно, але так, щоб було наочно продемонстровано коректність розрахунку результатів.

Аналітичні завдання функціональної залежності, ідентифікатори завдання додаткового дослідження функції та рекомендовані значення відповідних параметрів згідно з варіантом наводяться у таблицях 4.1, 4.2.

Таблиця 4.1 – Додаткові дослідження функцій

Літерний ID	Опис дослідження
1	2
#A	Є деяке число d. Визначити кількість значень функції, що були отримані в процесі табуляції, які є більшими від числа d, та розрахувати їх відсоткове співвідношення до загальної кількості
#B	Є деяке число d. Визначити кількість значень функції, що були отримані в процесі табуляції, які є меншими, ніж число d, та розрахувати їх відсоткове співвідношення до загальної кількості

Продовження таблиці 4.1

1	2
#C	Визначити кількість пар значень функції $\{f(x_{i+1}), f(x_i)\}$, де x_{i+1}, x_i є сусідні відліки аргументу функції, що були отримані в процесі табуляції, для яких виконується нерівність $f(x_{i+1}) > f(x_i)$, та розрахувати їх відсоткове співвідношення до загальної кількості
#D	Визначити кількість пар значень функції $\{f(x_{i+1}), f(x_i)\}$, де x_{i+1}, x_i є сусідні відліки аргументу функції, що були отримані в процесі табуляції, для яких виконується нерівність $f(x_{i+1}) < f(x_i)$, та розрахувати їх відсоткове співвідношення до загальної кількості
#E	Визначити кількість значень функції, що були отримані в процесі табуляції, для яких виконується нерівність $f(x_i) > x_i$, та розрахувати їх відсоткове співвідношення до загальної кількості
#F	Визначити кількість значень функції, що були отримані в процесі табуляції, для яких виконується нерівність $f(x_i) < x_i$, та розрахувати їх відсоткове співвідношення до загальної кількості

Таблиця 4.2 – Варіанти завдання 4.1

Варіант	Функція	ID	Рекомендовані значення параметрів
1	2	3	4
1	$y = \dfrac{\cos bx}{\frac{1}{2} + \sin^2 ax}$	#C	$x1 = 0$; $x2 = 3$; $\Delta x = 0{,}1;\ 0{,}25;\ 0{,}5$; $a = 2$; $b = 1{,}5$.
2	$y = \dfrac{\operatorname{arctg} bx}{a + \sin^2 x}$	#A	$x1 = 1$; $x2 = 4$; $\Delta x = 0{,}1;\ 0{,}2;\ 0{,}25$; $a = 1$; $b = 2$; $d = 1$.
3	$y = \dfrac{\ln x + \sin^2 ax}{b + \sqrt{x}}$	#B	$x1 = 0{,}5$; $x2 = 4$; $\Delta x = 0{,}2;\ 0{,}3$; $a = 2$; $b = 2$; $d = 0{,}4$.
4	$y = \dfrac{\ln^2(x+b)}{a + \sqrt{x+1}}$	#D	$x1 = 0$; $x2 = 2{,}5$; $\Delta x = 0{,}1;\ 0{,}2;\ 0{,}25$; $a = 2$; $b = 0{,}25$.

Продовження таблиці 4.2

1	2	3	4
5	$y = \frac{e^{-bx}}{b + \cos^2 ax}$	# E	$x1$= -1; $x2$ = 2; Δx = 0,2; 0,25; a = 2; b = 2.
6	$y = \frac{x^2 - 10}{b + \cos^2 ax}$	#C	$x1$= 0; $x2$ = 2,5; Δx = 0,1; 0,2; a = 2; b = 2.
7	$y = \frac{\cos bx}{\frac{1}{4} + \sin^2 ax}$	# E	$x1$= 0; $x2$ = 3,5; Δx = 0,2; 0,25; a = 2; b = 2.
8	$y = \frac{5 - \sqrt{ax}}{b + \mathrm{tg}^2 x}$	#B	$x1$= 0,5; $x2$ = 4,5; Δx = 0,2; 0,25; a = 2; b = 5; d = 0,25.
9	$y = b + a\cos x + \sin^2 x$	#D	$x1$= 0; $x2$ = 5; Δx = 0,25; 0,5; a = 1; b = 2.
10	$y = \frac{\ln^2(a + x)}{(b + x)^2}$	#C	$x1$= -0,5; $x2$ = 1,5; Δx = 0,1; 0,2; a = 1; b = 1.
11	$y = \frac{2 + \sin^2 ax}{b + \sqrt{x}}$	# F	$x1$= 0,5; $x2$ = 2,5; Δx = 0,1; 0,2; 0,25; a = 1; b = 1.
12	$y = b + \sin^2 x - \cos^2 ax$	#D	$x1$= 0; $x2$ = 2,5; Δx = 0,2; 0,25; 0,5; a = 2; b = 3.
13	$y = b - a\sin^2 x - \cos^2 x$	#B	$x1$= 0; $x2$ = 2,5; Δx = 0,2; 0,25; a = 2; b = 3; d = 1,5.
14	$y = \frac{\cos^2 x - a}{bx}$	# F	$x1$= 0,5; $x2$ = 3; Δx = 0,1; 0,2; 0,25; a = 2; b = 2.
15	$y = \frac{\cos bx}{\frac{1}{3} + \sin^3 ax}$	#A	$x1$= 0; $x2$ = 1,5; Δx = 0,1; 0,2; a = 2; b = 2; d = 0.

4.4 Розгляд основних пунктів виконання завдання 4.2

Мета цього завдання — дати його виконавцю навички організації в сценаріях VBScript ітераційних циклічних обчислень шляхом застосування оператора організації циклів за умовою DO…LOOP.

Розглянемо такий приклад варіанта завдання 4.2. Треба обчислити й вивести на екран шляхом складання мовою VBS сценарію та оформлення відповідного HTML-документа декілька значень такої функції:

$$\operatorname{arctg} x = \frac{\pi}{2} + \sum_{n=0}^{\infty} \frac{(-1)^{n+1}}{(2 \cdot n + 1) \cdot x^{2 \cdot n+1}} = \frac{\pi}{2} - \frac{1}{x} + \frac{1}{3 \cdot x^3} - \frac{1}{5 \cdot x^5} + \ldots$$

при $x > 1$.

Функція задана аналітичним виразом та за допомогою ряду Тейлора. Наближені значення функції через обчислення ряду потрібно зробити при різних значеннях точності ε: 0,00001; 0,0000001. Водночас необхідно рахувати кількість членів ряду, які були підсумовані до зупинки розрахунку. Критерієм зупинки обчислення ряду вважається виконання нерівності

$$\left| \frac{(-1)^{n+1}}{(2 \cdot n + 1) \cdot x^{2 \cdot n+1}} \right| < \varepsilon .$$

Для перевірки якості наближеного обчислення треба також сценарієм запровадити обчислення тієї самої функції при тих самих значеннях аргументу, але стандартними для мови VBS засобами.

Схема алгоритму розв’язання цього завдання подана на рисунку 4.2).

Теоретичний аналіз постановки завдання. Для цього варіанта завдання значення функції треба розраховувати для обмеженого діапазону значень аргументу: $1 < x < \pi/2$. Оскільки за умовою застосування ряду треба, щоб було значення аргументу $x > 1$, а за правилами застосування вбудованої функції Atn( ), яка обчислює функцію арктангенса, треба, щоб вхідні значення були в межах $-\pi/2$ та $\pi/2$.

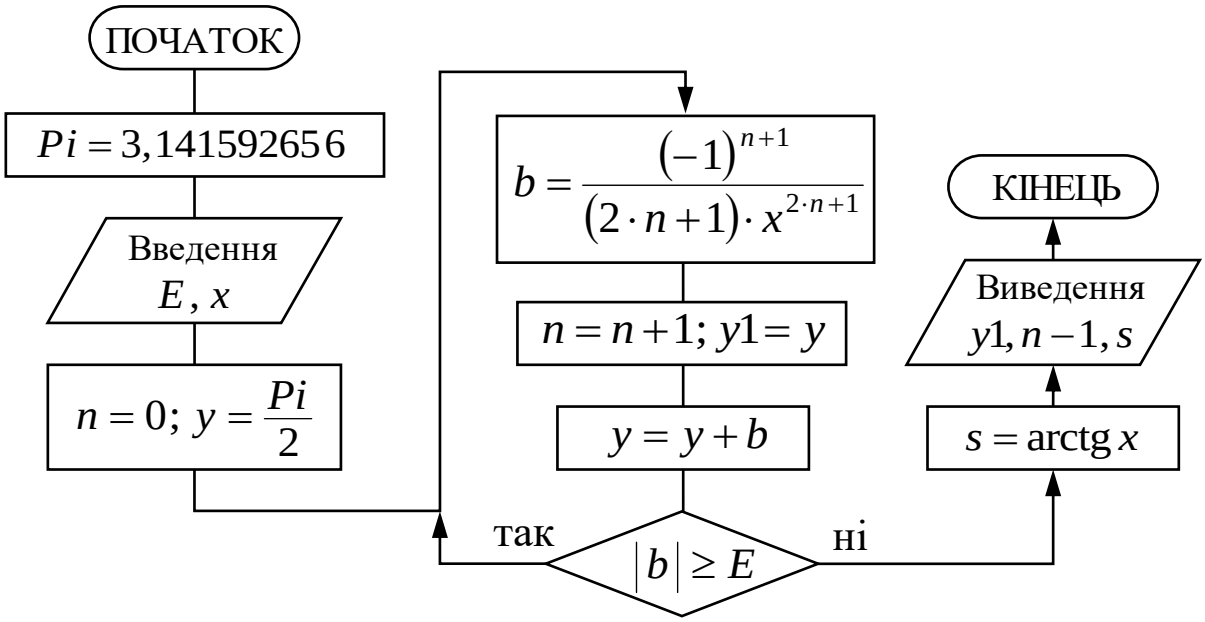


Рисунок 4.2 – Схема алгоритму розв’язання
прикладу завдання 4.2

Для перевірки правильності роботи сценарію розробляти контрольний приклад не треба, оскільки сценарій двічі обчислює те саме значення різними способами. Якщо ці значення збігаються (з певною похибкою), то це свідчить про коректність роботи коду.

Код HTML-документа зі сценарієм розв’язання наведеного прикладу може бути таким:

```
<HTML> <HEAD>
<TITLE>Застосування циклу Do...Loop (Завд. 4.2) </TITLE>
</HEAD> <BODY>
<H1>Обчислення функції за допомогою розкладання в ряд Тейлора</H1>
<P><INPUT TYPE="Text" ID="argumX" SIZE="10">
: значення аргументу функції<BR>
<INPUT TYPE="Text" ID="tochnE" SIZE="10">
: значення точності обчислення<BR>
<INPUT TYPE="Button" ID="knopka1"
VALUE="Обчислити!"></P>
<P><INPUT TYPE="Text" ID="rezultY" SIZE="16">
: значення функції, що обчислене за допомогою ряду<BR>
<INPUT TYPE="Text" ID="kilkistN" SIZE="10">
```

Продовження коду

```
: кількість членів ряду в сумі <BR>
        <INPUT TYPE="Text" ID="rezultS" SIZE="16">
: значення функції за стандартним способом обчислення
  <SCRIPT LANGUAGE="VBScript"> <!--
  DIM Pi
  Pi = 3.1415926536
  SUB knopka1_onclick
        DIM x, e, y, y1, n, s, b
        x = 0 + argumX.Value
        e = 0 + tochnE.Value
        n = 0
        y = Pi / 2
  DO
              b = (-1) ^ ( n + 1) / ((2 * n +1) * x ^ (2 * n +1))
              y1 = y
              n = n + 1
              y = y + b
        LOOP WHILE Abs( b ) >= e
        rezultY.Value = y1
        kilkistN.Value = n - 1
        rezultS.Value = Atn( x )
  END SUB
  --> </SCRIPT> </BODY> </HTML>
```

Завдання 4.2

Обчислити й вивести на екран шляхом складання мовою VBS сценарію та оформлення відповідного HTML-документа декілька значень функції, заданої за допомогою ряду Тейлора при різних значеннях точності ε. Також сценарій повинен давати змогу порівняти ці значення функції зі значеннями цієї ж функції, але обчисленими із застосуванням стандартних засобів, що має мова VBScript. У сценарії при обчисленні функції через ряд потрібно ще рахувати кількість членів ряду, які були підсумовані до зупинки розрахунку.

Дослідження провести з такими значеннями точності ε: 0,00001; 0,0000001. Значення функції виконавець завдання обирає самостійно.

Необхідні вхідні дані вводяться у середовище обчислень із клавіатури через елементи управління HTML, спосіб виведення розрахованих даних виконавець обирає самостійно. Варіанти завдання 4.2 наведено у таблиці 4.3.

Таблиця 4.3 – Варіанти завдання 4.2

Варі-ант	Завдання функції аналітичне та у вигляді ряду	Умови застосування
1	2	3
1	$\ln\left(\frac{x+1}{x-1}\right)=2\cdot\sum_{n=0}^{\infty}\frac{1}{(2\cdot n+1)\cdot x^{2\cdot n+1}}=2\cdot\left(\frac{1}{x}+\frac{1}{3\cdot x^3}+\frac{1}{5\cdot x^5}+\ldots\right)$	$\lvert x\rvert>1$
2	$e^{-x}=\sum_{n=0}^{\infty}\frac{(-1)^n\cdot x^n}{n!}=1-x+\frac{x^2}{2!}-\frac{x^3}{3!}+\frac{x^4}{4!}-\ldots$	$\lvert x\rvert<\infty$
3	$\ln(x)=2\cdot\sum_{n=0}^{\infty}\frac{(x-1)^{2\cdot n+1}}{(2\cdot n+1)\cdot(x+1)^{2\cdot n+1}}=2\cdot\left[\frac{x-1}{x+1}+\frac{(x-1)^3}{3\cdot(x+1)^3}+\frac{(x-1)^5}{5\cdot(x+1)^5}+\ldots\right]$	$x>0$
4	$\ln(x)=\sum_{n=0}^{\infty}\frac{(-1)^n\cdot(x-1)^{n+1}}{n+1}=(x-1)-\frac{(x-1)^2}{2}+\frac{(x-1)^3}{3}-\ldots$	$1<x\le 2$
5	$\ln(x)=\sum_{n=0}^{\infty}\frac{(x-1)^{n+1}}{(n+1)\cdot(x+1)^{n+1}}=\frac{x-1}{x}+\frac{(x-1)^2}{2\cdot x^2}+\frac{(x-1)^3}{3\cdot x^3}+\ldots$	$x>\frac{1}{2}$
6	$\ln(x+1)=\sum_{n=0}^{\infty}\frac{(-1)^n\cdot x^{n+1}}{n+1}=x-\frac{x^2}{2}+\frac{x^3}{3}-\frac{x^4}{4}+\ldots$	$-1<x\le 1$
7	$\ln\left(\frac{1+x}{1-x}\right)=2\cdot\sum_{n=0}^{\infty}\frac{x^{2\cdot n+1}}{2\cdot n+1}=2\cdot\left(x+\frac{x^3}{3}+\frac{x^5}{5}+\ldots\right)$	$\lvert x\rvert<1$
8	$\ln(1-x)=-\sum_{n=1}^{\infty}\frac{x^n}{n}=-\left(x+\frac{x^2}{2}+\frac{x^4}{4}+\ldots\right)$	$-1\le x<1$
9	$e^{-x^2}=\sum_{n=0}^{\infty}\frac{(-1)^n\cdot x^{2\cdot n}}{n!}=1-x^2+\frac{x^4}{2!}-\frac{x^6}{3!}+\frac{x^8}{4!}-\ldots$	$\lvert x\rvert<\infty$
10	$\cos x=\sum_{n=0}^{\infty}\frac{(-1)^n\cdot x^{2\cdot n}}{(2\cdot n)!}=1-\frac{x^2}{2!}+\frac{x^4}{4!}-\frac{x^6}{6!}+\ldots$	$\lvert x\rvert<\infty$
11	$\frac{\sin x}{x}=\sum_{n=0}^{\infty}\frac{(-1)^n\cdot x^{2\cdot n}}{(2\cdot n+1)!}=1-\frac{x^2}{3!}+\frac{x^4}{5!}-\frac{x^6}{7!}+\ldots$	$\lvert x\rvert<\infty$
12	$\sin x=\sum_{n=0}^{\infty}\frac{(-1)^{n-1}\cdot x^{2\cdot n-1}}{(2\cdot n-1)!}=x-\frac{x^3}{3!}+\frac{x^5}{5!}-\ldots$	$\lvert x\rvert<\infty$

Продовження таблиці 4.3

1	2	3
13	$\mathrm{e}^{x}=\sum\limits_{n=0}^{\infty}\frac{x^{n}}{n!}=1+\frac{x}{1!}+\frac{x^{2}}{2!}+\frac{x^{3}}{3!}+\frac{x^{4}}{4!}+\ldots$	$\lvert x\rvert<\infty$
14	$a^{x}=\sum\limits_{n=0}^{\infty}\frac{(x\cdot\ln a)^{n}}{n!}=1+\frac{x\cdot\ln a}{1!}+\frac{(x\cdot\ln a)^{2}}{2!}+\frac{(x\cdot\ln a)^{3}}{3!}+\ldots$	$\lvert x\rvert<\infty$
15	$\frac{\mathrm{e}^{x}-\mathrm{e}^{-x}}{2}=\sum\limits_{n=1}^{\infty}\frac{x^{2\cdot n-1}}{(2\cdot n-1)!}=x+\frac{x^{3}}{3!}+\frac{x^{5}}{5!}+\frac{x^{7}}{7!}+\ldots$	$\lvert x\rvert<\infty$

СПИСОК ЛІТЕРАТУРИ

1 Філіппенко І. Г., Гончаров В. О., Меркулов В. С. Основи алгоритмізації: конспект лекцій з дисципліни «Обчислювальна техніка та програмування». Харків: УкрДАЗТ, 2005. Ч. 2. 54 с.

2 Основи програмування інженерно-технічних розрахунків алгоритмічною мовою високого рівня: введення до VBSCRIPT: конспект лекцій з дисциплін «Обчислювальна техніка та програмування», «Інформатика» / С. Є. Бантюков, В. Г. Пчолін, І. Г. Бізюк, Р. О. Яровий, О. В. Казанко. Харків: УкрДУЗТ, 2020. 87 с.

3 Бутенко В. М., Павленко Є. П., Головко О. В. Інженерія програмного забезпечення. WEB-програмування: навч. посіб. Харків: УкрДУЗТ, 2019. 99 с.

4 Основи алгоритмізації базових обчислювальних процесів: навч. посіб. / Данько М. І., Меркулов В. С., Бутенко В. М., Бізюк І. Г., Головко О. В., Гончаров В. А. Харків: УкрДАЗТ, 2008. 163 с.

5 Глушаков С. В., Жакин И. А., Хачиров Т. С. Программирование Web-страниц / шеф-ред. С. В. Глушаков. Харьков: Фолио, 2005. 390 с.

МЕТОДИЧНІ ВКАЗІВКИ
до виконання лабораторних робіт

з дисциплін
«ОБЧИСЛЮВАЛЬНА ТЕХНІКА І ПРОГРАМУВАННЯ»
та *«ІНФОРМАТИКА»*

Частина 1

Відповідальний за випуск Бантюков С. Є.

Редактор Еткало О. О.

Підписано до друку 15.06.21 р.

Формат паперу 60х84 1/16. Папір писальний.

Умовн.-друк.арк. 3,5. Тираж 5. Замовлення №

Видавець та виготовлювач Український державний університет
залізничного транспорту,
61050, Харків-50, майдан Фейєрбаха, 7.
Свідоцтво суб'єкта видавничої справи ДК № 6100 від 21.03.2018 р.